HIGH LEVEL DECOMPOSITION FOR BIPEDAL LOCOMOTION PLANNING

A Dissertation
Presented to
The Academic Faculty

By

Michael X. Grey

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy in
Robotics

Georgia Institute of Technology

August 2017

HIGH LEVEL DECOMPOSITION FOR BIPEDAL LOCOMOTION PLANNING

Approved by:

Dr. Liu, C. Karen
School of Interactive Computing
Georgia Institute of Technology

Dr. Ames, Aaron D.
Mechanical and Civil Engineering
& Control and Dynamical Systems
California Institute of Technology

Dr. Egerstedt, Magnus
School of Electrical and Computer Engineering
Georgia Institute of Technology

Dr. Hauser, Kris
Department of Electrical and Computer Engineering
& Department of Mechanical Engineering and Materials Science
& Department of Computer Science
Duke University

Dr. Zucker, Matt
Engineering Department
Swarthmore University

Date Approved: June 20, 2017

I dedicate this thesis to Mike Stilman—an incredible roboticist who we lost far too soon.

ACKNOWLEDGEMENTS

First, I would like to extend my deepest appreciation to my advisers, Karen Liu and Aaron Ames, for swooping in and helping me to get my research efforts on track during a time in my life that was riddled with uncertainty. Their guidance and support has been instrumental in seeing this thesis through to the end. I am also grateful to Magnus Egerstedt, Kris Hauser, and Matt Zucker, for serving on my reading committee.

Furthermore, I would like to thank Annie Antón and Andrea Thomaz for all of the logistical support that they provided to ensure that my graduate career could continue without interruption. It was a strange, tragic, and difficult transition, but their unwavering support ensured that I could keep moving forward.

Thanks to all the members of the Humanoid Robotics Lab: Ana Huamán, Can Erdogan, Kyle Volle, Neil Dantam, Tobias Kunz, Martin Levihn, Jon Scholz, Saul Reynolds-Haertle, and Heni Ben Amor. We were like a small family—a family based around the shared goal of getting robots to do *really cool stuff*.

To those who stood with me during the most daunting technical challenge that we might ever face, I extend my deepest thanks. Pete Vieira, Andrew Price, Eric Huang, Sungmoon Joo, and Matt Zucker: our camaraderie during the DARPA Robotics Challenge was genuinely the most rewarding aspect of the whole experience.

To my friends Taylor, Ben, Amber, and Tiffany, who provided me with some semblance of a life outside of the lab, I thank you all for the wonderful memories, and I look forward to making more in the future.

To my family, and especially my parents, thank you for all your love and support. You always encouraged me to pursue my dreams, and I know you are happy to see that I have done exactly that.

And finally, I would like to thank my first adviser, Mike Stilman. Without him, I might not have seriously considered pursuing humanoid robotics research, but Mike was an in-

spiration and a remarkable enabler. He will be dearly missed.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

SUMMARY

Legged robotic platforms offer an attractive potential for deployment in hazardous scenarios that would be too dangerous for human workers. Legs provide a robot with the ability to step over obstacles and traverse steep, uneven, or narrow terrain. Such conditions are common in dangerous environments, such as a collapsing building or a nuclear facility during a meltdown. However, identifying the physical motions that a legged robot needs to perform in order to move itself through such an environment is particularly challenging. A human operator may be able to manually design such a motion on a case-by-case basis, but it would be inordinately time-consuming and unsuitable for real-world deployment.

This thesis presents a method to decompose challenging large-scale motion planning problems into a high-level planning problem and a set of parallel low-level planning problems. We apply the method to quasi-static bipedal locomotion planning. The method is tested in a series of simulated environments that are designed to reflect some of the challenging geometric features that a robot may face in a disaster scenario. We analyze the improvement in performance that is provided by the high- and low-level decomposition, and we show that completeness is not lost by this decomposition.

CHAPTER 1
INTRODUCTION

Legged Locomotion

Mobile robot platforms are poised to have a transformative effect on human society due to their potential for automating a wide variety of physical labor and performing dangerous tasks without risk to human lives. Traditional mobile robot platforms depend on a set of wheels whose axles are fixed with respect to the chassis of the robot. Such systems tend to have severely limited mobility in an unaccommodating environment, because even a small obstacle can be an insurmountable barrier. By comparison, legged robotic platforms have broader mobile capabilities, because they can reconfigure the way they contact their environment in order to overcome obstacles.

Interest in legged robotic platforms has been growing as they become an increasingly viable technology. Potential applications include household assistance [1], search and rescue [2], disaster relief [3], humanitarian aid (such as defusing mines) [4], as well as science and exploration in dangerous areas on Earth [5] and all the way into outer space [6, 7, 8].

A variety of mechanical designs have been conceived to carry out the broad range of potential applications. Bipedal robots are often designed to reflect human proportions, with the aim of allowing these robots to function in environments originally designed for humans [1, 9, 10, 11], although not all bipedal robots resemble humans [12, 13]. Another popular design is quadrupedal robots, which tend to resemble dogs or high-agility animals like cheetahs [14, 15, 16]. More legs might also be used [4, 6, 8] for enhanced stability, often at the cost of increasing the mechanical complexity and occupying more space. Some platforms may also leverage the advantages of both legs and traditional wheels by incorporating wheels into their legged designs [17, 18].

Figure 1.1: Robots with various mobility characteristics. Left: PR2 by Willow Garage, a traditional mobile manipulator. Right: Atlas by Boston Dynamics, a bipedal humanoid. Bottom: RoboSimian by NASA's Jet Propulsion Laboratory, a quadruped with wheels on its knees and feet.

However, before the value of these designs can be realized in practice, algorithms are needed that can tap into the physical capabilities of the platforms. Conventional teleoperation methods tend to be too limited or burdensome for human operators to use while contending with the complexity of coordinating all the robot's moving parts. These robots need to maintain balance and avoid collisions while performing their tasks, which requires a level of precision that cannot be expected of a human operator. Offloading the cognitive burden of making decisions about motions from the human operator onto autonomous algorithms would make the operation of legged robots more tenable in practice.

Multi-Stage Planning

To improve the autonomy of complex robotic systems, we can turn to planning. The field of planning aims to produce strategies or sequences of actions that satisfy a set of constraints while accomplishing a goal. In the context of robot planning, these constraints usually emerge from the robot's initial state, the physical model of the robot and its environment, and the objectives given to the robot. The goal of motion planning in particular is to generate sequences of feasible joint motions that can bring the robot from a start configuration to a goal configuration. With a motion planning method that is able to utilize the full kinematic capabilities of a legged platform, we can make the deployment of legged robots more practical while tapping into their full potential.

However, planning methods tend to struggle with problems that are both high-dimensional and heavily constrained. Legged robots tend to require many joints, making them high-dimensional systems. In order to successfully move themselves, legged robots also need to keep their balance, maintain environmental contact, and avoid collisions, making them heavily constrained. Even worse, legged robots are hybrid continuous-discrete dynamic systems which further complicates the matter since hybrid planning tends to be considerably more challenging than purely continuous or purely discrete planning.

A common approach for making this type of planning problem tractable is to break

it down into stages or hierarchies. This is especially common when combining task and motion planning, which is another type of high-dimensional, heavily-constrained, hybrid continuous-discrete planning problem [19, 20, 21, 22]. This approach has also been applied to legged locomotion planning [23, 24, 25, 26].

By creating abstractions of the planning problem, it is possible to quickly solve high-level aspects of the problem before needing to address the computationally expensive low-level details where all of the dimensions and constraints of the problem need to be considered. Decomposing the problem into these high-level and low-level stages allows us to solve problems that might otherwise be intractable, while still utilizing the full physical capabilities of the robot.

Thesis Statement

Autonomous motion planning offers an effective way to utilize the full kinematic capabilities of a robot platform and to maximize the value of deploying robots in challenging environments. The more autonomy that can be achieved in determining a robot's actions, the less burden is suffered by human operators and engineers; this creates a lower barrier to real-world deployment and a higher ceiling of effectiveness for robots.

We can achieve greater levels of autonomy by decomposing the planning problem into high- and low-level stages. If the high-level stage can quickly evaluate and resolve the big-picture problem, then it can provide guidance to the low-level stage, resulting in faster convergence towards finding a solution.

Overview of Thesis

This thesis presents three related contributions to the field of motion planning:

- The *Randomized Possibility Graph* (RPG) which is a formalization of route-based motion planning methods [27],

- a proof-of-concept implementation of the RPG, applied to bipedal humanoid motion planning on arbitrary terrain, and
- a theoretical proof of probabilistic completeness, as well as a convergence analysis which predicts that the RPG can improve the likelihood of finding a solution under certain conditions [28]. Those conditions are provided by the analysis.

The next chapter provides a brief overview of prior approaches used for legged locomotion planning, with an emphasis on the particular methods that this thesis will build upon. Chapter 3 describes the Randomized Possibility Graph (RPG) concept, which is the central contribution of this thesis. Chapter 4 details how the RPG can be applied to bipedal locomotion planning. Chapter 5 examines the probabilistic completeness of the RPG when applied to semi-unstructured environments. Finally, Chapter 6 concludes the work and discusses potential future applications for the RPG.

CHAPTER 2
RELATED WORK

A variety of algorithms for locomotion planning have been developed in the last few decades, each with distinct strengths and weaknesses. In this section, we will review some of the most popular algorithms pertaining to legged locomotion planning with an emphasis on the concepts which this thesis will build upon. We begin with the highest-level approach to locomotion planning and then end with the lowest-level approaches, reflective of how the RPG reasons about the overall planning problem.

Route Planning

Prior to the development of legged robots, mobile manipulation platforms would use motorized wheels in order to move through their environments. This form of locomotion would usually be holonomic, lending itself nicely to basic randomized sampling methods, like Probabilistic Roadmaps (PRM) [29] and Rapidly-exploring Random Trees [30]. Legged locomotion is a heavily constrained process wherein the feasibility constraint manifold is a lower dimension than the robot's configuration space; as a result, basic randomized sampling methods cannot directly handle the problem. However, certain simplifications to the dynamics model can approximate legged locomotion as a low-dimensional holonomic process, allowing basic randomized sampling methods to be applied. We refer to such high-level approaches as *route planning*.

One way to approach route planning is to cast the entire walking process as holonomic by enveloping all possible walking motion within a bounding box. This can be done in a two-stage planner [23] where the collision geometry of the lower body is encapsulated in a bounding cylinder while the upper body collision geometry is ignored. PRM or RRT methods can be used on the bounding cylinder to find a route for the robot. The fact

that the cylinder is collision-free guarantees that the agent can follow the path without its legs colliding with the environment. The second stage of the planner accounts for the upper body by adjusting the trajectories of the arm joints to avoid obstacles that were not accounted for by the bounding cylinder. The naive use of a bounding box means that this approach is not complete with respect to the full kinematics of the robot. Using a solid cylinder around the entire lower body is especially problematic since it will not allow the robot to step over any obstacles or change elevation.

The bounding box concept can be modified to use multiple bounding boxes [31, 32] where one bounding box can encapsulate the upper body while each foot is given an independent bounding box. The bounding box for the upper body is lifted above the ground while the bounding boxes for each foot reach down to the ground to ensure that the randomly sampled path has suitable footholds. This decoupling between bounding boxes enables the planner to find ways to step over 3D obstacles on the ground. However, it is still unable to utilize the full kinematics of the robot, because the bounding box of the upper body overestimates the necessary clearance. As an example, the planner could not figure out how to duck under an overhanging obstacle, even if the robot is capable of doing so.

Bounding box route planning methods are inherently limited in their ability to utilize the full physical capabilities of the platform, because they rely on *inner* approximations of the robot's constraint manifold. Instead, it is possible to create an *outer* approximation which overestimates what the robot can do. For example, it is possible to reduce the collision geometry to just the root geometry and use a randomized sampling method to find a path that allows the root geometry to move from a start to a goal configuration like the reachability-based planner of Tonneau, et. al. [33]. The reachability-based planner also takes into account whether the environment provides a set of reachable contact points that are adequate for static equilibrium along the route. Since a large set of contacts is often available, a heuristic is used downselect. After a route is found, Tonneau, et. al. attempt to interpolate motions along the route which move the robot's hands and feet between the

contacts identified along the route. However, since the feasibility of the route is based on an outer approximation, there is no guarantee that the interpolation will work. In the case of a failure, the process is redone from scratch with the hope that the randomization will succeed on the next attempt.

It is possible to use both an inner and an outer approximation as guidance within a single planning problem. This is shown by Shimizu and Sugihara [34]. Depending on which approximation is satisfied, the high-level planner can query an appropriate planner, whether that be a footstep planner or a multi-modal motion planner. We formalize and generalize this idea via the Randomized Possibility Graph [27].

Footstep Planning

Some of the earliest work in legged locomotion planning used dynamic programming on a search tree with a discrete set of primitive stepping motions [35]. When a robot stands on one foot, there is a continuous region of feasible foot placements relative to the stance foot that the other foot (the swing foot) can reach. However, taking a discrete set of samples from this region, as shown in Fig. 2.1, and limiting the tree search to use these candidate foot placements allows a finite branching factor when planning forward over the possible actions. The feasible whole body trajectories for each candidate footstep can be precomputed, allowing the search tree to be expanded without any expensive whole body inverse kinematics operations.

This approach is able to generate footstep plans quickly enough for online use [36]. It can also be applied to environments with stairs [24]. This approach is complete with respect to the discretized action set. It is also globally optimal with respect to arbitrary cost functions when an admissible heuristic is used to guide an A* search [24]. Route planning methods that used an outer estimate of the system dynamics could be used in a heuristic to guide the search.

However, this discretized approach is not optimal nor complete with respect to the full

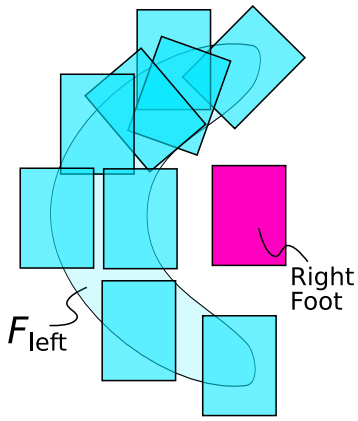


Figure 2.1: Example of the feasible left step region, F_{left}, being discretized by a set of samples.

kinematics of the robot platform. Using a discrete set of primitive steps inherently limits how the robot may utilize its kinematics to navigate around obstacles. Some augmented versions of the classic footstep planning approach include an adaptive local search to supplement the discrete branching [37]. This way, when the terrain is pathological enough that the discretized set of footsteps is insufficient to find a feasible footstep sequence, the planner can do a local search to find feasible footholds. This allows a wider range of feasible steps to be accounted for while still maintaining a finite branching factor. Despite being adaptive, it is still not complete with respect to the full kinematic capabilities of the robot, because it does not search for paths through the full configuration space of the robot.

Multi-modal Motion Planning

Instead of simplifying the planning problem with bounding boxes or a discrete set of action primitives, it is possible to decompose the problem based on "modes". In [25, 38], a *mode* is defined by a fixed set of contact points between the robot and its environment. Each unique mode has its own feasibility constraint manifold. When it is possible for the constraint manifolds of two different modes to intersect, those modes are considered *adjacent*. There are two prominent Multi-modal Motion Planning methods presented by Hauser et. al.: Mutli-modal PRM [25] and Random-MMP [39].

The basic workflow of Multi-modal PRM is to first sample *transition configurations*—which lie in the intersections of adjacent modes—and then find feasible paths through each individual mode that can link together those transition configurations. When finding paths through the individual modes, a constrained PRM is constructed for each mode, hence the name Multi-modal *PRM*. The modes themselves are identified by searching for feasible combinations of contact points between the robot and the environment. Traditionally, the points on the robot and the environment that are allowed to be used for contact are predetermined and provided as input to the planner, or the environment is constructed in a way that allows more contact points to be sampled over time (for example, using a height map). The combinatorial complexity of all the different modes that could be considered may become enormous and not scale well to difficult problems. This motivates the use of Incremental-MMPRM which focuses the search over a small selection of candidate modes. When the set of candidate modes are exhausted, the set is expanded. In the worst case scenario, this behavior degrades into the basic MMPRM algorithm. The candidate modes can be selected in a variety of ways, for example if a set of modes lead from the start to the goal via mode-adjacency *and* each adjacent pair contains a feasible transition configuration, then the set of modes would be a promising choice. This heuristic is referred to as *Search Among Feasible Transitions*.

A more broadly applicable Multi-modal Motion Planning algorithm is Random-MMP. This algorithm draws its inspiration from RRT-based methods by building trees rather than building a graph like PRM. Random-MMP is a more general algorithm which applies to any problem where new modes can be sampled within the proximity of an existing mode. Similar to RRT, Random-MMP can grow trees by taking uniformly random samples of states (and occasionally a sample from the goal, to improve performance) and extending the existing tree towards that state, performing any necessary mode transitions along the way. A more ideal growth behavior would be based on the Expansive Space Tree (EST) [40, 41] method. In this ideal method, samples are taken uniformly from the locally reachable

space instead of uniformly over the entire state space. This allows the trees to be grown more reliably instead of frequently running into infeasible states.

Both of these Multi-modal Motion Planning methods have the characteristic of being probabilistically complete. This is an important feature for allowing the planners to solve problems that are arbitrarily complex or difficult, and it is the key advantage that these methods have over all other approaches. However, an important disadvantage of these methods is performance. Depending on the difficulty of the problem, the run-time may be on the order of minutes or hours, in contrast to most locomotion planning methods which aim to run at a real-time rate.

CHAPTER 3
RANDOMIZED POSSIBILITY GRAPH

For a robot to perform useful work autonomously in a complex environment, it must be able to determine ways that it can move itself from a start configuration to a goal configuration without violating the laws of physics or inflicting damage on itself or its environment. These requirements that are placed on the robot are known as feasibility constraints. The space of all configurations which satisfy the constraints is called the feasibility constraint manifold. For complex robotic systems, this manifold tends to be lower-dimensional than the configuration space and is often defined implicitly rather than explicitly, making it difficult to know its exact geometry.

Motion planning methods ordinarily operate by constructing graphs or trees which consist of configurations that fully exist within the feasibility constraint manifold of the action they are performing. Remaining within this manifold is a reasonable requirement to place on the graph, because any vertices or edges which lie outside of the manifold are, by definition, invalid—which may mean it is physically impossible, or simply harmful to the robot or its surroundings. Unfortunately, for a humanoid robot to remain on the constraint manifold, expensive calls to a whole body inverse kinematics (IK) solver must be performed [42, 43, 44]. This results in a critical bottleneck if a broad area needs to be explored before finding a solution.

The key idea of the Randomized Possibility Graph (RPG) is to explore the *possibility* of an action first, instead of immediately committing to costly whole body inverse kinematics queries. Therefore, the problem is broken down into two stages: A high-level stage which explores possibilities $\mathcal{P}$, and a low-level stage which attempts to generate motions $\mathcal{M}$. The high-level graph generated during $\mathcal{P}$ will guide the efforts of the low-level planners used by $\mathcal{M}$.

The governing logical principles behind the RPG have a theoretical grounding in Possibility Theory [45], but the concepts are intuitive enough that a knowledge of Possibility Theory is not necessary to proceed. It is enough to understand that the *possibility* of any given action e can be labelled with "impossible", "possible", or "indeterminate" depending on whether it satisfies the necessary conditions (C_N) or the sufficient conditions (C_S) that are assigned to it:

$$e.\text{label} = \begin{cases} \text{“impossible”} & \text{if } C_N(e) \text{ is false} \\ \text{“possible”} & \text{if } C_S(e) \text{ is true} \\ \text{“indeterminate”} & \text{otherwise} \end{cases} \tag{3.1}$$

If we design necessary and sufficient conditions that can be checked much more quickly than querying the original constraint manifold, we can then construct a *Randomized Possibility Graph*, whose states (vertices) are connected by either "possible" or "indeterminate" edges, and expand it very efficiently for whole body motion planning.

Sufficient vs. Necessary Conditions

To construct the RPG, we must first design sufficient and necessary conditions for the feasibility constraint manifold of the action whose possibilities we are exploring. We should design the conditions to be quick to test in order to reduce the computational cost of exploration. The conditions should also use as few parameters as is reasonable, because randomized search methods tend to be more effective in low-dimensional search spaces.

Suppose we have a 2D constraint manifold, C, which exists in a 3D state space (Fig. 3.1). Let the xy-plane be a low-dimensional feature space which contains the essential information for navigating C. We denote the projection of C by C_P. Even with a flattened-out projection, identifying which points are inside or outside of the manifold may still be costly or difficult, because the boundary of C_P may consist of functions that are expensive

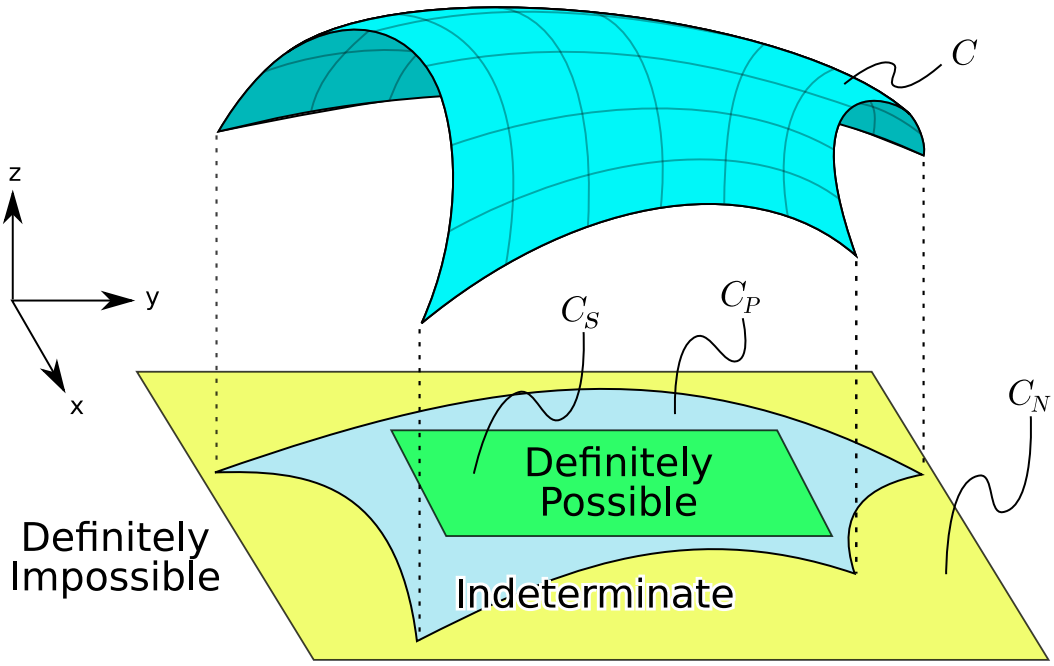


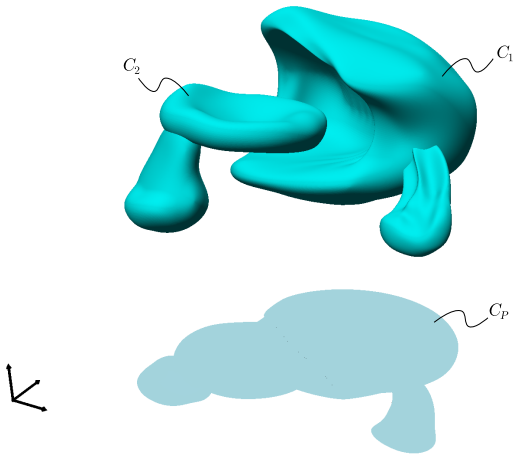
Figure 3.1: Visual depiction of an abstract constraint manifold, C and its projection. The manifold is projected, C_P, from 3D space onto a plane. "Sufficient" C_S and "Necessary" C_N boundaries are fitted within and around the projection of the manifold. Elements inside the green box are definitely "Possible". Elements outside of the yellow box are definitely "Impossible". Elements inside the yellow box but outside of the green box are "Indeterminate" because they might or might not lie on the projection. Identifying whether a point lies inside or outside of C_S or C_N may be considerably faster than identifying whether it lies inside or outside of C_P.

to compute or hard to fully define. However, suppose a box, circle, or some other simple shape can be fit within C_P such that it is *guaranteed* that every point within the simple shape also lies within the manifold projection. Such a shape would be a suitable representation of the sufficient condition manifold, C_S. Any point lying inside of C_S also lies inside of C_P and should be labelled with "possible". Similarly, if C_P can be bounded by a simple shape, C_N, such that $C_P \subseteq C_N$, then C_N would qualify as the necessary condition manifold. Any point lying outside of C_N should be labelled with "impossible". Finally, any point inside of C_N but outside of C_S should be labelled with "indeterminate".

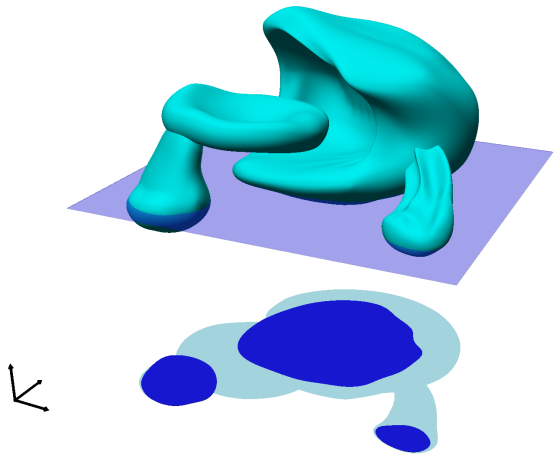
Higher Dimensions

The systems we are interested in have feasibility constraint manifolds higher than 2. As a result, there are additional considerations that need to be taken into account. For example, in Figure 3.2a the feasibility constraints form two disconnected manifolds, C_1 and C_2, but their projections overlap to form C_P. When constructing C_S for this scenario, we need to guarantee that every pair of connected points in C_S can map to a pair of connected points in C_1 or C_2. To ensure this, we choose a slice through the constraint manifolds and consider its projection as shown by the blue shadow in Figure 3.2b. The manifolds determined by the sufficient conditions must then fit inside the projection of this slice.

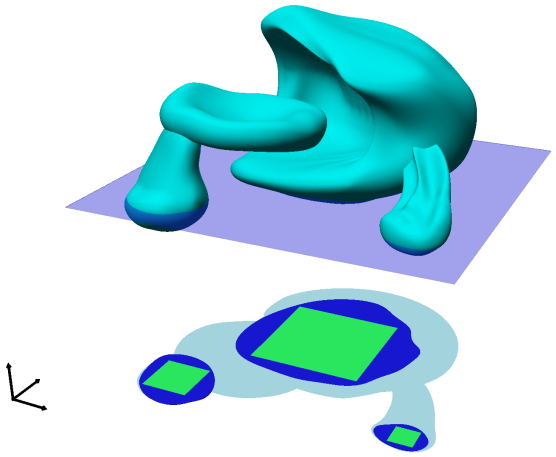
On the other hand, the necessary conditions do not need any additional consideration when used in higher dimensional scenarios. The only strict requirement for the necessary conditions is that they envelop the entire projection of the constraint manifold. In some cases, such as Figure 3.2d, this may mean that two connected points in C_N map to disconnected points in C. Whenever possible, this should be avoided, perhaps by changing the necessary condition function or changing the space that the constraint manifolds are being projected to.



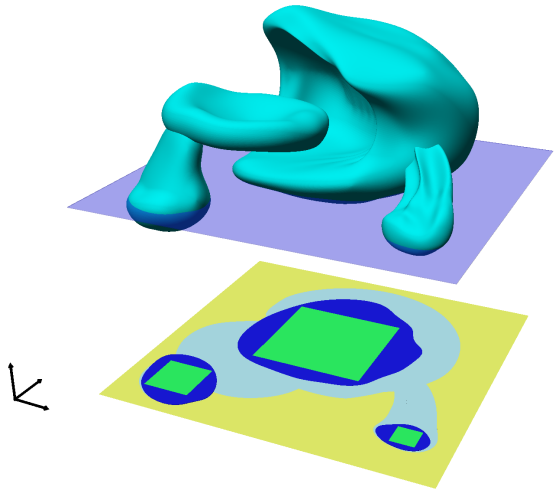
(a) Disconnected manifolds in higher dimensions might produce a fully connected projection.



(b) We can project a slice of the higher dimensional manifold such that points that are connected within the projection can map to points that are connected within the higher dimensional manifold.



(c) The sufficient conditions must be designed to fit within the projections that map to full connected subsets of the constraint manifolds.



(d) The necessary conditions must still envelop the entire projection C_P.

Figure 3.2: In higher dimensions, additional considerations are needed when constructing sufficient conditions.

Mapping from Low to High

Given a set of sufficient conditions that corresponds to connected subsets within the feasibility constraint manifolds, any low-dimensional points which satsify the sufficient conditions can be mapped directly to predetermined high-dimensional points within the feasibility constraint manifolds. Paths between these connected low-dimensional points are guaranteed to map to paths within the high-dimensional feasibility constraint manifold, as depicted in Figure 3.3. While the low-dimensional points in the sufficient condition manifolds may be able to map to an infinite continuum of other points in the feasibility constraint manifolds, the other high-dimensional points are not important for navigation, which is the central purpose behind the low-dimensional approximations. If the robot's goal configuration is outside of the constraint manifold subset that the sufficient conditions map to, then this will be accounted for during the low-level planning stage discussed in the next section.

Low-dimensional points that only satisfy the necessary conditions are not as convenient. We do not immediately know what high-dimensional points they might map to, or whether such high-dimensional points even exist. To accommodate challenging scenarios, we consider a region around each of these indeterminate low-dimensional points instead of focusing on the exact points themselves. These regions are extruded into cylinders as illustrated in Figure 3.4, and we sample points within the intersection of the cylinder and the feasibility constraint manifold. Due to the ambiguity of whether these sampled points will be able to connect to each other, we expect to take an arbitrary number of samples until paths can be found or a resource limit is reached. Unlike the sufficient conditions, we do not know a priori whether a path will exist between these sampled points. The next section discusses how we search for paths between these points.

Probabilistic Conditions

For heavily constrained high-dimensional systems, a low-dimensional projection might wash away a considerable amount of information that would be relevant for guarantee-

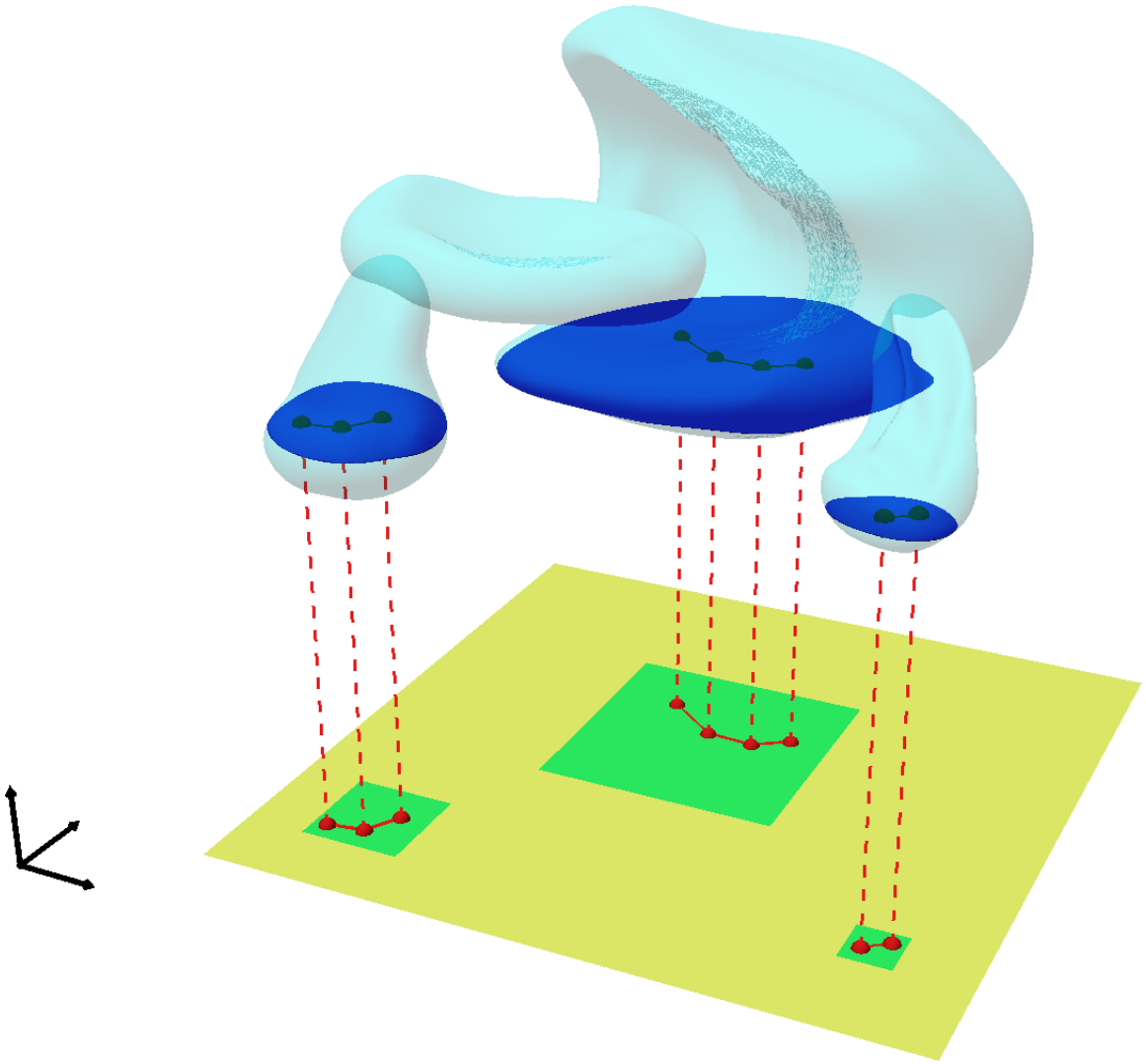


Figure 3.3: Connected points within each sufficient condition manifold can map directly to connected points in the full feasibility constraint manifolds.

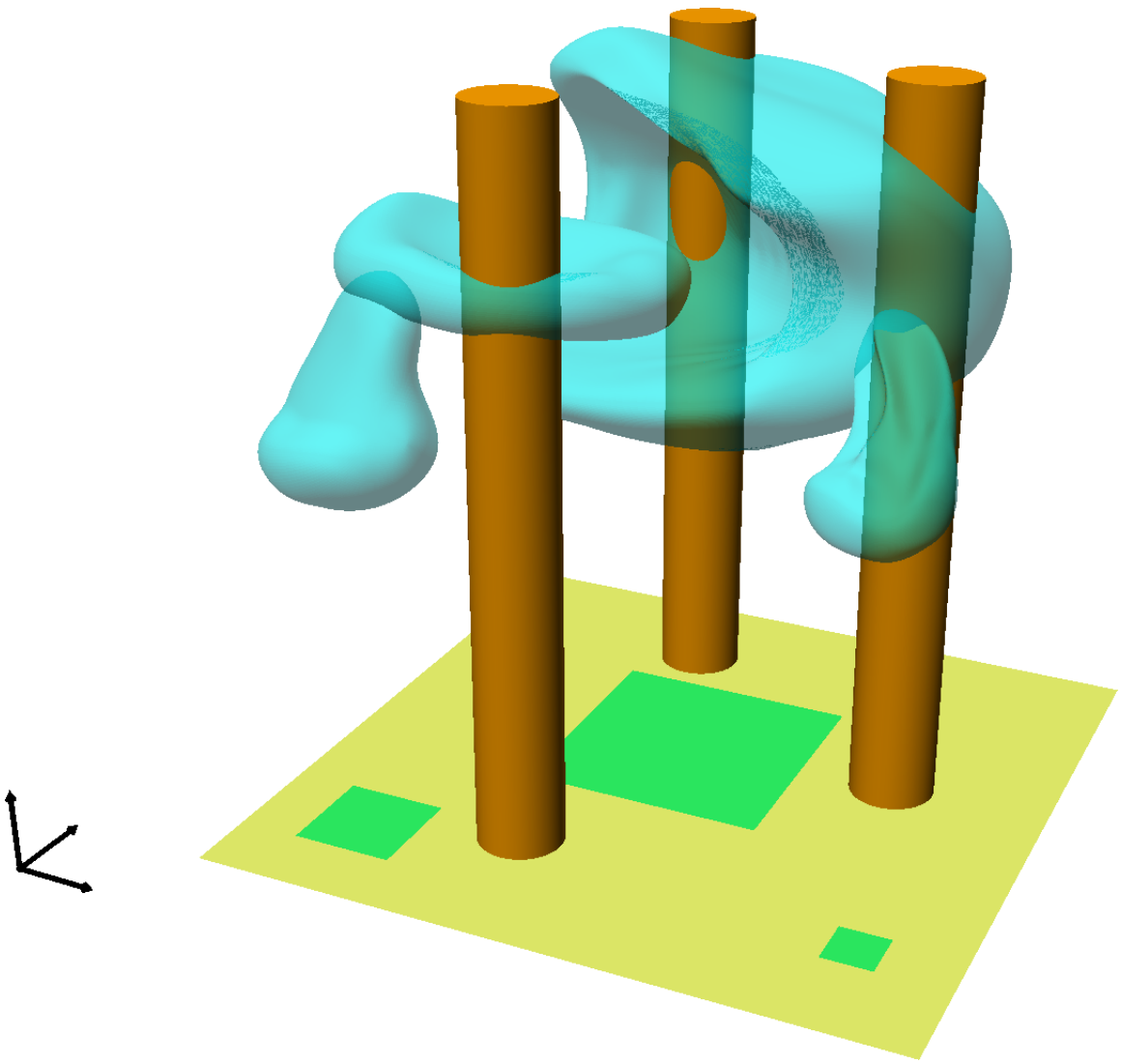


Figure 3.4: When mapping from the necessary condition manifold to the full feasibility constraint manifolds, we consider a region around the low-dimensional point and extrude a cylinder (orange) into the high-dimensional space, then attempt to sample points within the intersection of the feasibility constraint manifolds and the cylinder.

ing feasibility of a route. There may be fine details in the environment that would strongly impact feasibility, but these details would not be captured by a projection which is low-dimensional enough to be suitable for defining a route. For example, the terrain in some region of the map might not offer adequate footing for a walking robot, but this would go unnoticed if the necessary conditions operated solely on the transformation of the robot's root link.

To overcome this limitation, we can design probabilistic conditions which utilize randomized sampling to test the necessary or sufficient conditions in a higher-dimensional parameter space than what the low-level route exists in. We can search for values for additional parameters by taking samples within range limitations and using gradient descent methods when available. For example, the necessary conditions might sample the ground for feasible foot placements within a reachable region of the robot's root transform. If a set of values is found that satisfies the condition, then the high-level search can proceed. If no set of values is found within a conservative number of attempts, then the high-level search should abandon extending that route and attempt to extend a different route.

Since the values of the additional parameters are at least partially determined through randomness, their satisfaction is probabilistic rather than deterministic. A route which failed to satisfy a probabilistic necessary condition may still be revisited later if the high-level route planner randomly selects it again for expansion. As explained later in Section 3.2, a randomized sampling planner will be used for the high-level exploration. When choosing which existing route to extend towards a new location in the environment, the high-level planner does not take into consideration any of the "additional" parameters that are sampled by any probabilistic conditions. In other words, the high-level planner is not concerned with whatever foot placement locations may have been sampled along each existing route; it only cares about the distances within the low-dimensional projection in which the routes are defined. Therefore, we exclude the additional parameters from the data structure of the high-level graph. However, an efficient implementation may cache these

values elsewhere for later use in the low-level stage, which is described in Section 3.2.

RPG Definition and Construction

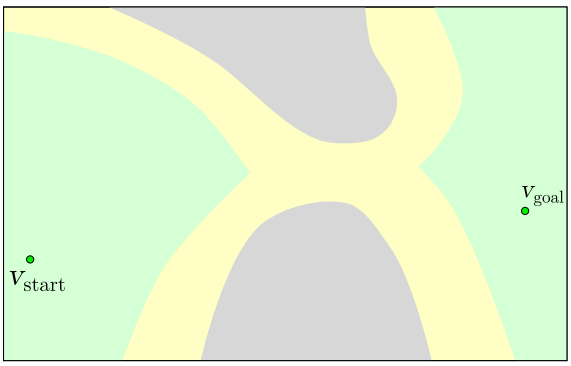
In the previous section, we introduced the concepts of C_N and C_S, the necessary and sufficient (respectively) condition manifolds which occupy a lower dimensional space than the state space. We define $\mathscr{E} = P(X)$ to be the "possibility exploration space" where X is the state space of the robot, and in this paper $P : X \rightarrow$ SE(3) maps from the robot's state to the SE(3) transformation of the robot's root link. In general, P is chosen to be a projection operator such that $C_S \subseteq C_N \subseteq P(X)$. We will use $\mathscr{E}$ as the search domain for the possibility exploration stage $\mathcal{P}$.

Definition 1. A Randomized Possibility Graph is a tuple

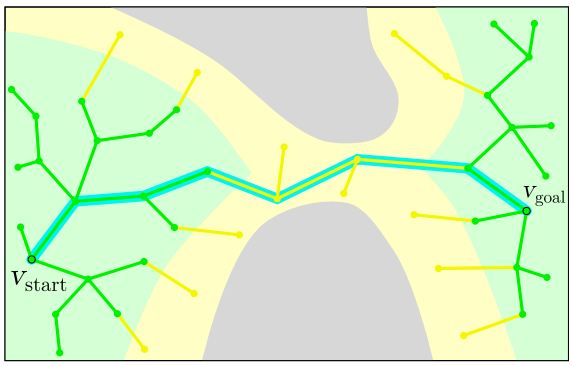
$$RPG = (\Gamma_{\mathcal{P}}, \Phi_{\mathcal{M}}, \Omega_{\mathcal{M}}, \Gamma_{\mathcal{M}})$$

where,

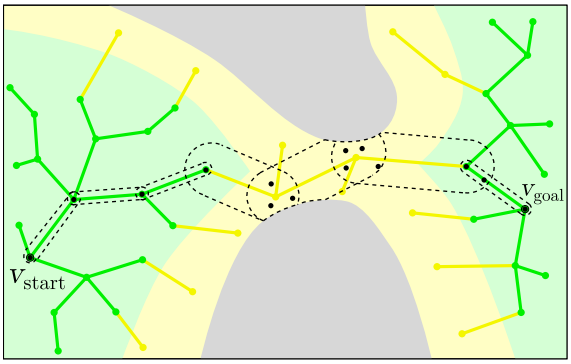
- $\Gamma_{\mathcal{P}} = (V_{\mathcal{P}}, E_{\mathcal{P}})$ is a graph where $V_{\mathcal{P}}$ is a set of vertices which are elements of $\mathscr{E}$, and $E_{\mathcal{P}}$ is a set of directed edges, each with a "possible" or "indeterminate" label,
- $\Phi_{\mathcal{M}} : E_{\mathcal{P}} \times E_{\mathcal{P}} \rightarrow X^k$ is an operator which takes two edges and produces a set of $k > 0$ states,
- $\Omega_{\mathcal{M}} : X \times E_{\mathcal{P}} \times X \rightarrow X^f$ is an operator which maps two states with a possibility edge in between them into a discretized trajectory of $f \geq 0$ states, where $f = 0$ implies failure to find a trajectory,
- $\Gamma_{\mathcal{M}} = (V_{\mathcal{M}}, E_{\mathcal{M}})$ is a graph where $V_{\mathcal{M}}$ is a set of vertices which are elements of X, and $E_{\mathcal{M}}$ is a set of directed edges which indicate feasible paths between the vertices of $V_{\mathcal{M}}$.



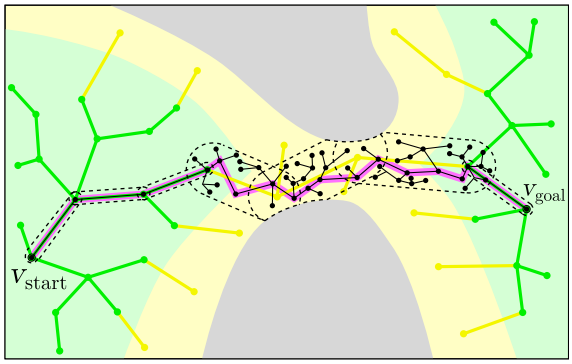
(a) The RPG is initialized with just a start vertex and goal vertex. The pale green regions are areas where C_S is satisfied, pale yellow is where C_N is satisfied, and gray is where C_N is violated. These colored regions represent information which is not directly known to the algorithm, so it must be searched via randomized sampling.



(b) A randomized sampling motion planner has constructed $\Gamma_{\mathcal{P}}$ to find a path from the start to the goal through $\mathscr{E}$. This provides a guide route, which is highlighted in cyan.



(c) The dotted regions represent the ranges of π_S or π_N for each edge along the guide route. The π_S regions are thin because we are guaranteed to find a solution directly along the routes where C_S is satisfied. Conversely, π_N may need to search a broader area to find a solution. $\Phi_{\mathcal{M}}$ is applied to the edges of the guide route, generating sets of states (black dots) where the ranges of the planners overlap.



(d) States and whole body paths (black dots and edges) are generated by $\Omega_{\mathcal{M}}$ using the guide route and the states generated by $\Phi_{\mathcal{M}}$. These elements are put into $\Gamma_{\mathcal{M}}$ until a complete whole body path is found from the start to the goal.

Figure 3.5: Illustration of the RPG procedure

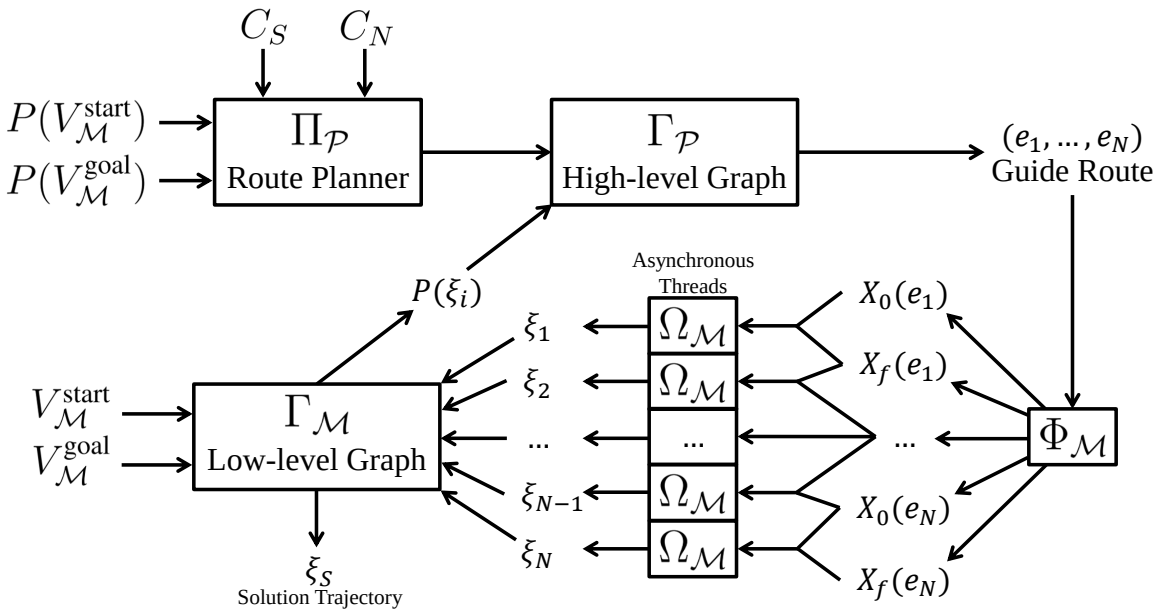


Figure 3.6: Workflow of a Randomized Possibility Graph. At the top is Stage $\mathcal{P}$ which generates a Guide Route. The Guide Route is then fed through the Stage $\mathcal{M}$ which is shown on the bottom. Stage $\mathcal{M}$ decomposes the Guide Route into a set of low-level planning problems which run in parallel. The results of the low-level planning queries are aggregated in $\Gamma_{\mathcal{M}}$, and their projections are passed back up to $\Gamma_{\mathcal{P}}$ to assist in identifying more routes if a solution has not been found yet.

The graph $\Gamma_{\mathcal{P}}$ is used to solve the first stage, $\mathcal{P}$, which explores possibilities. $\Gamma_{\mathcal{P}}$ can be constructed using a sample-based motion planner, such as PRM [29] or RRT [30]. If C_N or C_S have a small volume within $\mathscr{E}$, then a projection-based sampler may be needed, such as CBiRRT [46] where C_N and C_S are treated as task constraints. Alternatively, C_N and C_S can be treated as "hard" and "soft" task constraints respectively (meaning that C_N is required but C_S is preferred) which would make the method in [47] more applicable. For this paper, we use the "hard" and "soft" constraint approach, and we refer to this high-level planner as $\Pi_{\mathcal{P}}$.

We initiate $\Gamma_{\mathcal{M}}$ with one or more "start" states, $V_{\mathcal{M}}^{\text{start}}$, and one or more "goal" states, $V_{\mathcal{M}}^{\text{goal}}$. $\Gamma_{\mathcal{P}}$ is initiated with the projections of these states, $P(V_{\mathcal{M}}^{\text{start}})$ and $P(V_{\mathcal{M}}^{\text{goal}})$. The high-level motion planner of choice, $\Pi_{\mathcal{P}}$, is used to find a route $(e_1, ..., e_n \in E_{\mathcal{P}})$ which connects a start projection to a goal projection while remaining within C_N. The solved path generated by $\Pi_{\mathcal{P}}$ represents a guide route, similar to the "guide trajectories" of [33]. An illustration of this process can be seen in Figs. 3.5a-3.5b. The guide route is used to focus the efforts of the next stage, $\mathcal{M}$.

Provided the guide route found by $\mathcal{P}$, we want to use $\Omega_{\mathcal{M}}$ to examine each segment of the route to see if it can be realized in X. We define $\Omega_{\mathcal{M}}$ as:

$$\Omega_{\mathcal{M}}(x_i^0, e_i, x_i^f) = \begin{cases} \pi_S(x_i^0, e_i, x_i^f) & \text{if } C_S(e_i) \\ \pi_N(x_i^0, e_i, x_i^f) & \text{if } C_N(e_i) \wedge \neg C_S(e_i) \end{cases}$$

where π_S and π_N are sub-planners which produce full state space trajectories given a sub-start state (x_i^0), a sub-goal state (x_i^f), and an edge (e_i) of $E_{\mathcal{P}}$ which is used as a "guide". The "guide" edge may be used to compute a heuristic, determine footstep locations, or focus randomized samples, depending on the nature of the planner. We choose π_S such that it is guaranteed to quickly find a solution along routes where the sufficient conditions C_S are satisfied. π_N is a whole body motion planner, ideally with a probabilistic completeness guarantee. π_N can be applied to edges which satisfy the necessary conditions C_N even if

the sufficient conditions are not satisfied, but it is not guaranteed to return a solution.

To find a feasible continuous motion through state space, we need to choose x_i^f and x_{i+1}^0 to be equal. Define $X_0(e_i)$ to be the set of states that can be used as sub-start states, x_i^0, for $\Omega_{\mathcal{M}}(x_i^0, e_i, x_i^f)$, and $X_f(e_i)$ to be the set that can be used as sub-goal states, x_i^f. We can then define $\Phi_{\mathcal{M}}(e_i, e_{i+1}) = X_f(e_i) \cap X_0(e_{i+1})$. Using $\Phi_{\mathcal{M}}$ to determine the endpoints (sub-starts and sub-goals) used by $\Omega_{\mathcal{M}}$ ensures that the RPG is able to create continuous state trajectories. Note that $\Phi_{\mathcal{M}}(e_0, e_1) = V_{\mathcal{M}}^{\text{start}} \cap X_0(e_1)$, and $\Phi_{\mathcal{M}}(e_n, e_{n+1}) = X_f(e_n) \cap V_{\mathcal{M}}^{\text{goal}}$. $\Phi_{\mathcal{M}}$ is illustrated by the black dots in the overlapping dotted regions of Fig. 3.5c. Determining $X_0(e)$ and $X_f(e)$ will depend on the implementation of π_S and π_N, but most planners have either a discrete set of permissible endpoints or a continuous set that can be sampled from. In practice, many planners allow multiple start and goal states to be specified per query.

The overall procedure for planning with the RPG is to generate a guide route $(e_1, ..., e_n \in E_{\mathcal{P}})$ by constructing $\Gamma_{\mathcal{P}}$ with $\Pi_{\mathcal{P}}$ and feeding that route through $\Phi_{\mathcal{M}}$ and then through $\Omega_{\mathcal{M}}$. Whenever $\Omega_{\mathcal{M}}$ identifies feasible state trajectories, the states and edges of those trajectories are added to $\Gamma_{\mathcal{M}}$. A solution is found when $\Gamma_{\mathcal{M}}$ contains a path from a start to a goal state. The procedure is illustrated in Figure 3.5 and diagrammed in Figure 3.6.

Depending on the implementation of π_N, it may take an indeterminable amount of time to produce a solution. Moreover, it might not be able to produce a solution for some e_i if the edge is not a truly feasible guide route. Rather than waiting for $\Omega_{\mathcal{M}}$ to return a result of success or failure, the stage $\mathcal{P}$ can search for alternative guide routes by deleting any of the indeterminate edges of the guide route from $\Gamma_{\mathcal{P}}$ and then continuing to grow $\Gamma_{\mathcal{P}}$ in parallel to $\mathcal{M}$. This parallelism allows the RPG to avoid being bottlenecked by challenging routes when alternatives exist. When elements are added to $\Gamma_{\mathcal{M}}$, their projections can be added to $\Gamma_{\mathcal{P}}$ as "possible" elements to assist the ongoing high-level search.

CHAPTER 4
BIPEDAL LOCOMOTION PLANNING

In this chapter, we detail how the RPG concept can be applied to bipedal locomotion planning. Stage $\mathcal{P}$ requires a set of sufficient conditions and a set of necessary conditions which can be inspected at a high level while searching for a route. These conditions must be outer and inner (respectively) approximations of the robotic system's feasibility constraints.

Once a candidate route is discovered, stage $\mathcal{M}$ requires a low-level planning routine that can examine segments of the route and return feasible whole body motions whenever a route is viable. The low-level planner should take advantage of the full configuration space available to the robot in order to ensure that none of the robot's kinematic capabilities are overlooked.

Designing High-level Conditions

To design the high-level conditions, we first consider the physical feasibility constraints of the problem. For this work, we limit the scope of the problem to only use quasi-static (as opposed to dynamic) motions. This leaves us with the following set of feasibility constraints:

- Avoid collisions
- Maintain quasi-static balance
- Enforce kinematic constraints to maintain contact points
- Remain within joint torque and position limits

In this section, we discuss how to simplify these constraints into a set of sufficient conditions and a set of necessary conditions which can be applied to bipedal locomotion

across arbitrary terrain. In Section 3.2 we introduce the concept of the possibility exploration space, $\mathscr{E}$, the space that gets explored in order to find a possible route.

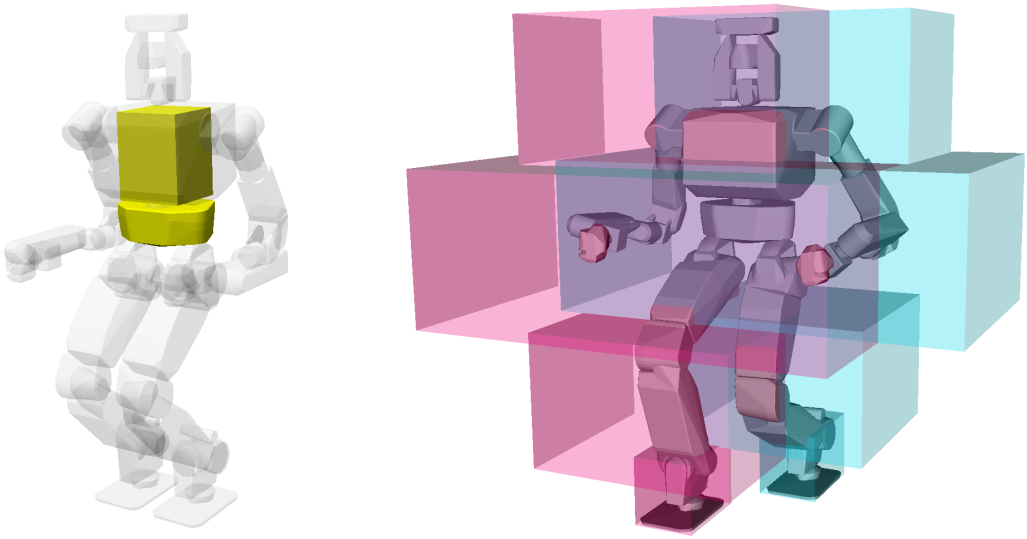
The choice of what constitutes $\mathscr{E}$ may vary between the problems that the RPG is being applied to. For bipedal locomotion planning, we choose $\mathscr{E}$ to be the SE(3) transform of the robot's root link. Therefore, the conditions must be designed so that they can be tested based on the root link transformation of the robot.

For the conditions to have the best possible effect, the sufficient conditions should be an inner approximation of the constraint manifold that is as loose as possible while still guaranteeing the viability of the route through $\mathscr{E}$. The necessary conditions should be an outer approximation of the constraint manifold that is as restrictive as possible while still guaranteeing that they do not rule out regions of $\mathscr{E}$ that are actually viable.

Collision Avoidance Conditions

Most robots in operation today—especially legged robots—are actuated with strong motors, controlled by a stiff PD control loop. If an unintentional collision occurs between one of the robot's links and the environment, this is almost certain to result in severe damage, either to the robot or to the environment: either by breaking the object that the robot collides with, by causing the robot to topple over, or by burning out a motor due to excessive internal torquing. Even if the robot uses a compliant controller when colliding with objects, the unexpected external force could throw it off balance, or the very presence of the obstacle could prevent the robot from reaching its goal.

The collision geometry of a robot is ordinarily defined by a set of meshes or geometric primitives which are specified per each rigid body that belongs to the robot. The positions and orientations of these rigid bodies (such as the foot, torso, hand, head, and everything in between) are a function of the robot's full set of joint positions, plus the SE(3) transformation of the root link (in our case, the robot's pelvis). Clearly this means that robot's collision geometry is a function of all the robot's joint positions, plus root transform.



(a) Minimal geometry used for necessary conditions (Yellow).

(b) Expanded geometries used for sufficient conditions. Each foot has a set of geometries attached to it (blue for the left and pink for the right), allowing these conditions to work on uneven terrain, and to step over small obstacles.

Figure 4.1: Collision geometries used for the (a) necessary and (b) sufficient conditions. Since (b) occupies more space, the constraint manifold for the sufficient conditions is smaller—and therefore more restrictive—than the manifold for the necessary conditions.

Since $\mathscr{E}$ is defined by *only* the transform of the root link, the necessary conditions for collision avoidance should only consider the subset of the overall collision geometry that is a function of the root transform, but otherwise invariant for all joint positions. This ensures that the necessary condition does not rule out any routes that may be viable by using some non-standard set of joint positions. At the same time, it ensures that the planner does not waste effort on any clearly impossible routes, such as a route that passes through a wall or an excessively thin passage.

The collision geometry used for the sufficient conditions must guarantee that consecutive vertices in $\Gamma_{\mathcal{P}}$ have enough free space that there is guaranteed to be a feasible path through configuration space that allows the root link to move from the SE(3) transform of one vertex to the next. To accomplish this, we use an "expanded" collision geometry similar to [31, 32, 48] where a set of inflated boxes are attached to each foot, illustrated in

Figure 4.1b. As long as a route edge contains a sequence of feasible alternating left-right-left foot placements where

- each subsequent foot placement is reachable from the previous, and
- these expanded collision geometries do not intersect the environment when positioned at each foot placement

then there is guaranteed to be a feasible walking trajectory along the route segment. This guarantee can be ensured by pre-determining a set of parameterized primitive trajectories which move exclusively within the bounds of the expanded collision geometries. Alternatively, the dimensions of the boxes can be generated to accommodate a set of parameterized primitive trajectories.

Note that this sufficient condition for collision avoidance presupposes that we have a sequence of foot placements sampled along the route edge. The procedure for sampling foot placements is discussed later in Section 4.2. When testing the possibility of an edge e, the operation $C_S(e)$ should sample foot placements along e and then store that information in a cache mapped to e to avoid repeated effort later on. Also note that even if two connected edges both satisfy C_S, their combined route might not necessarily satisfy C_S, because the foot placements leading up to their shared vertex might not be able to reach each other (although the probability of this is low). If such a case arises, the vertex where the edges meet can be treated as an infinitesimally short *indeterminate* edge and use stage $\mathcal{M}$ to find a whole body trajectory to connect the two edges.

To accommodate sloped terrain, the bottom segment of the expanded collision geometry can be given rotational joints that reflect the behavior of the robot's ankle joints as shown in Figure 4.2. This allows the expanded geometry to accurately evaluate the available free space above the foot placements.

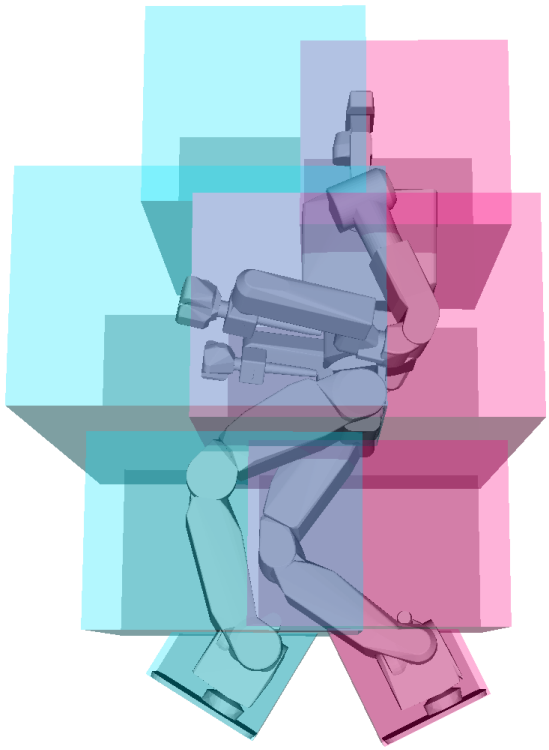


Figure 4.2: The bottom segment of each expanded collision geometry has a two-axis revolute joint which allows it to conform to slopes while the other segments of the geometry can remain upright.

Footstep Feasibility Constraints

The simplified collision geometries are effective for navigating around environmental obstacles, but they are not adequate as sufficient or necessary conditions on their own. If the floor of an environment has large gaps, clutter, or changes in elevation, then a collision-free path might still be infeasible due to an absence of viable foot placements. Here we discuss the conditions for a foot placement to be feasible, which include:

- Inclination limitations
- Support geometry
- Free space

Slope limitations are determined by the coefficient of friction between the robot's foot

sole and the region of contact. An in-depth analysis of how friction impacts static equilibrium for an arbitrary set of contact points can be found in [49]. However, since we are interested specifically in flat-footed bipedal robots, there are a number of simplifications which can be made. Bipedal robots transition between double-support phases (standing on two feet) and single-support phases (standing on one foot). During a single-support phase, the entirety of the robot's weight rests on one foot. Given a static coefficient of friction, μ_s, in order for static equilibrium to be maintained, it can be shown that the maximum inclination of the contacts is $\tan^{-1}(\mu_s)$. Any hypothetical foot placements whose inclination exceeds this value should be rejected.

Since the foot placements are to be use for quasi-static motions, we also need to consider constraints on their support geometries. Quasi-static motions require a support polygon with non-zero area, or else static stability cannot be ensured. The support polygon is the convex hull of the robot's contact points, projected down along the direction gravity. Note that in the general case, the full support *region* could extend beyond this projected polygon [49], but in the case of flat-footed bipedal walking the most constrained phases for the support region are the single-support phases, during which the support region *does* perfectly coincide with the traditional support polygon. Since the most constrained limitations on the support region must be satisfied for the overall problem to be satisfied, it is sufficient to use the traditional support polygon simplification with no loss of completeness.

If the support polygon has zero area, it would imply that the robot is dynamically balancing on a sharp edge or curved surface. However, quasi-static motion requires the robot to be statically stable at any given moment during the motion. Therefore, any hypothetical foot placements whose support area would be very small must be rejected. To encourage more stable foot placements, it is a good idea to initially inflate the lower bound on the required area and gradually reduce it if suitable foot placements are not being found.

Other characteristics of the support geometry can be considered, like the distance of the centroid from the closest edge. A larger distance from the nearest edge would correspond to

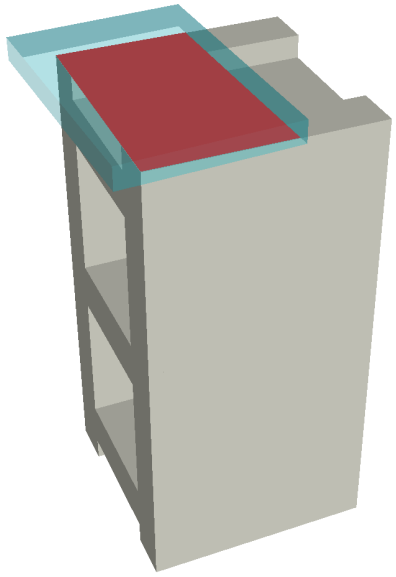


Figure 4.3: Support geometry (red) of a hypothetical foot placement (transparent blue) on top of a cinder block.

a more broadly distributed (and therefore more stable) support base. Figure 4.3 illustrates the support polygon of a single footstep.

While the bottom of the foot must have adequate contacts to support the weight of the robot, the rest of the foot's geometry must not intersect any of the environment's geometry. When testing sufficient conditions, the expanded geometries of Figure 4.1b should be used at each foot placement. For necessary conditions, only the collision geometries that are rigidly attached to the bottom of the foot should be tested, as illustrated in Figure 4.4. If these minimal foot geometries intersect the environment, then the foot placement is physically impossible, no matter what the configuration of the rest of the robot may be. Whether testing the necessary or sufficient conditions, the minimal collision geometries of sequential foot placements must not intersect each other, since that would be equivalent to a self-collision.

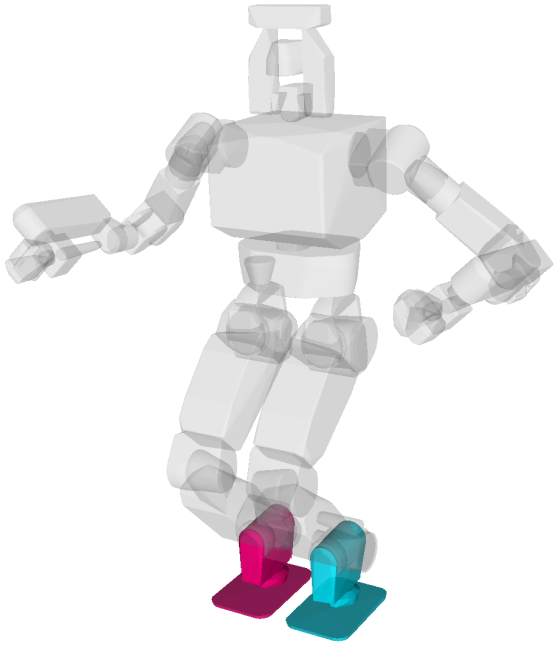


Figure 4.4: Minimal geometries of the left (blue) and right (pink) feet. A necessary condition for a hypothetical foot placement to be feasible is for these geometries to be collision-free when placed there.

Foot Placement Sampling

Given an arbitrary geometric representation of an environment, there may not be a closed-form expression for the locations of viable foot placements. To overcome this, we would like to employ a sampling method that can quickly produce a suitable discrete set of foot placements to consider while planning. Furthermore, we would like the sampling method to scale well to large environment models, so the samples should be concentrated in regions that are likely to assist in finding a solution. If samples are taken in regions of the environment which cannot be reached by the robot, then it would be a waste of computational time and resources.

In this section, we detail a method for sampling viable foot placements. The approach here builds on the reachability-based method described by [33]. The limitations of this method are discussed in Section 4.2.3.

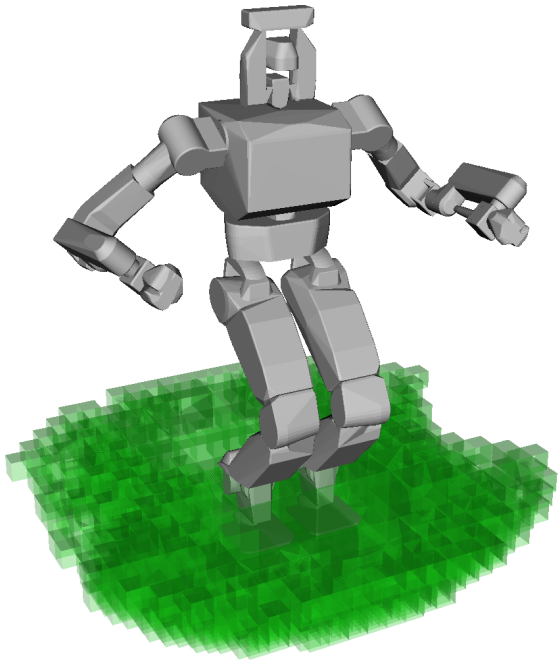


Figure 4.5: An approximation of the space that can be reached by either of the robot's legs using any foot rotation, represented by an octree. The approximation is an outer bound on the reachable space, meaning that there may be some space included in the octree which is not actually reachable, but none of the reachable space is left out. This ensures that we do not overlook any possible foot placements.

Reachability

The reachable space of a limb seldom has a closed-form representation, except for limbs with particularly simple kinematics. In the general case, there is no inherently obvious way to characterize a limb's reachable space. In order to search for feasible foot placements within an environment that is represented by a set of meshes or geometric primitives, we would like a geometric representation of the reachable space which we can check for intersections with the environmental geometries.

The geometric representation should include the entirety of the reachable space to ensure that our foot placement search does not diminish the overall completeness of our locomotion planner. Furthermore, we expect to do many thousands of searches for foot placements, so the representation should also allow for efficient collision checking. Using

an octree, like in the work of [33], would satisfy both of these requirements. An example illustration of an octree representation of the reachable space of a bipedal robot's legs can be seen in Figure 4.5.

If we use an RRT-based sampling method for the possibility exploration stage $\mathcal{P}$, then each time we extend the graph through $\mathscr{E}$, we will be extending from a location which is potentially reachable from the start or goal configurations. If we always extend from potentially reachable regions, then we can have some confidence that the new frontiers being explored may also be reachable from the start or goal.

Plane Sampling and Adjustment

Given a suitable representation of the reachable space, we can find sets of points, $P_{\text{intersection}}$, where the reachable space intersects with the geometry of the environment. The distribution of these points may vary depending on the layout of the environment, the geometric features, and the methods used for intersection detection. In our implementation, FCL [50] was used.

Figure 4.6 illustrates examples of what the distribution of intersection points between the reachable space octree and an environment may look like. It is worth noting that higher concentrations of intersection points do not correspond to greater availability of foot placements—in fact, more often the opposite relationship is true, since higher concentrations of intersections usually emerge from geometric features like curves and edges which are less likely to accommodate foot placements than flat ground. Therefore, it is important that our foot placement identification method does not get inadvertently biased towards searching around high concentrations of intersection points.

To ensure that the reachable space is searched fairly, take a point, p_{target}, within the reachable space and find the $N_{\text{neighbors}}$ closest intersection points, $P_{\text{neighbors}}$ (see Fig. 4.7b), where $N_{\text{neighbors}}$ is a user-defined value (in our implementation, we use 10). From $P_{\text{neighbors}}$, we randomly select sets of triples until one of the sets can define a suitable plane. For a

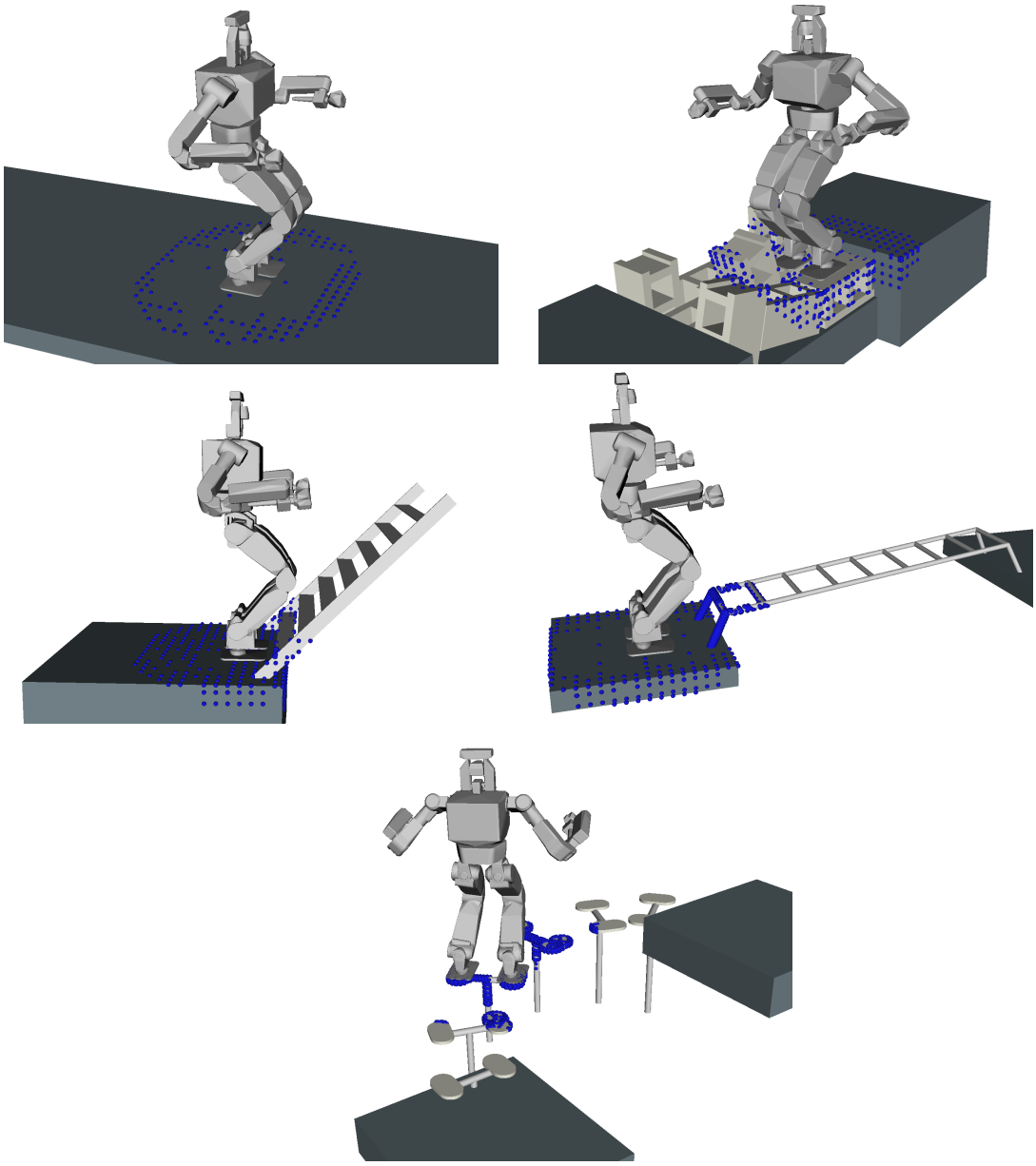


Figure 4.6: Examples of points which are found by intersecting the reachable space geometry (see Fig. 4.5) with various environment geometries. The intersection points are illustrated as small blue spheres. Depending on the layout and geometric representation of the environment, some regions might be densely packed with intersection points while other regions are sparsely packed. This variability in density is accounted for by sampling from planes determined by the intersection points instead of using the intersection points directly.

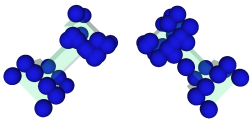
plane to be suitable, the three points that define it must not be collinear. Additionally, the plane's angle of inclination (the angle between the plane's normal vector and the direction of gravity) must not exceed $\tan^{-1}(\mu_s)$. The function `IsSuitablePlane` used in Algorithm 1, line 9 tests to ensure that the triplet of points is not collinear and that the plane they form does not exceed the maximum inclination.

Algorithm 1: Sampling foot placements. $\mathcal{R}$ is the reachable space described in Section 4.2.1. Env is the environment geometry. $\mathcal{F}$ is the simplified foot geometry.

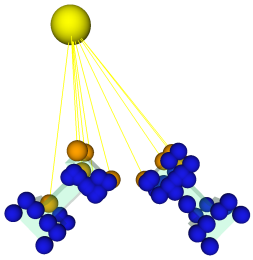
```
1  Function SampleFootPlacement(R, Env)
2      P_intersection ← SampleIntersection(R, Env);
3      for i in [1, N_SampleAttempts] do
4          p_target ← SampleRandomTranslation(R);
5          P_neighbors ← FindClosestNeighbors(P_intersection, p_target, N_neighbors);
6          for j in [1, N_PlaneAttempts] do
7              P_plane ← SampleTriplet(P_neighbors);
8              for k in [1, N_MaxIterations] do
9                  if not IsSuitablePlane(P_plane) then
10                     break;
11                 p_sample ← SampleRandomSE2(P_plane);
12                 if IsFeasiblePlacement(p_sample) then
13                     return p_sample;
14                 SetPosition(F, p_sample);
15                 P_collisions ← SampleIntersection(F, Env);
16                 if IsEmpty(P_collisions) then
17                     continue;
18                 p_swap ← GetFurthestFromPlane(P_collisions, P_plane);
19                 p_closest ← FindClosestNeighbor(P_plane, p_swap);
20                 Swap(p_swap, p_closest);
21     return null;
```

The function `SampleRandomSE2` randomly samples a SE(2) point on the plane such that the center of the foot placement is within the triangular boundary of the three intersection points—the rotation is randomly sampled from $[0, 2\pi)$ around the normal axis of the plane. This point defines a hypothetical foot placement on the plane. An illustration of the hypothetical placement sampling can be seen in Figure 4.7d.

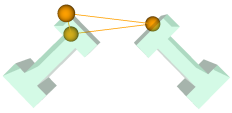
We check for collisions between the environment and a simplified foot geometry (a thin



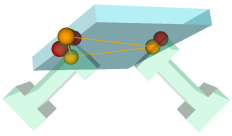
(a) Find intersection points between the environment and the reachable space.



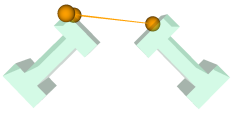
(b) Randomly sample a point within the reachable space and find a set of N closest intersection points.



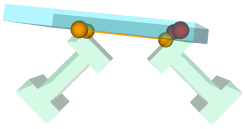
(c) Randomly select triplets out of the set of N closest points until one of the triplets provides a viable plane.



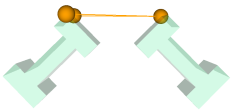
(d) Place a simple foot geometry randomly on the plane within the boundary of the three points. Check for collisions between this foot placement sample and the environment.



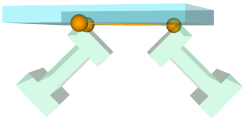
(e) Replace one of the plane points with the collision point (from the previous step) which is furthest from the plane.



(f) Again, sample a foot placement and check for collisions.



(g) Again, Replace one of the plane points with the furthest collision point from the previous step.



(h) Viable foot placement has been identified.

Figure 4.7: Illustration of the iteration process. The transparent green I-beams are a cross section of the environmental features being investigated. Blue spheres represent the intersection points between the environment and the reachable space. The yellow sphere is a point randomly sampled within the reachable space. Orange spheres are points that are being used to construct hypothetical planes. The blue box is a hypothetical foot placement sample.

rectangle) located at the hypothetical foot placement. If collisions exist, then we take the collision point which is furthest from the plane and use it to determine a new sampling plane. Whichever point from the previous plane is closest to the collision point is replaced by it, forming a new plane and triangle as seen in Figure 4.7e. This process is repeated until the simplified foot geometry no longer intersects the environment.

Once a collision-free plane is found, we continue to sample potential foot placements on the plane until one of those foot placements satisfies the feasibility constraints described in Section 4.1.2. This rejection sampling approach has shown to be effective on a diverse range of geometric features, but has limitations which will be discussed in Section 4.2.3.

Performance Enhancements

Using a purely random-sampling based approach is effective at identifying unanticipated foot placements, but an important disadvantage is that it tends to produce a sloppy and inconsistent pattern of samples. If a bipedal robot always walks using a strangely distributed set of foot placements, its walking behavior will be inefficient—not to mention, aesthetically unappealing.

Rather than relying only on random samples, it can be a good idea to seed the sampling with a few "ideal" positions, and the method can test if a viable foot placement is available near that position. To incorporate this into the algorithm, modify the function `SampleRandomTranslation` (Algorithm 1, line 4) to initially return the seeded positions before producing random samples.

Similarly, the function `SampleRandomSE2` (Algorithm 1, line 11) can be seeded to return positions that are close to the centroid of the P_{plane} triangle, which have a higher likelihood of providing an adequate support geometry. Additionally, biasing the samples to be oriented in the direction that the robot is facing may improve the usefulness of the samples.

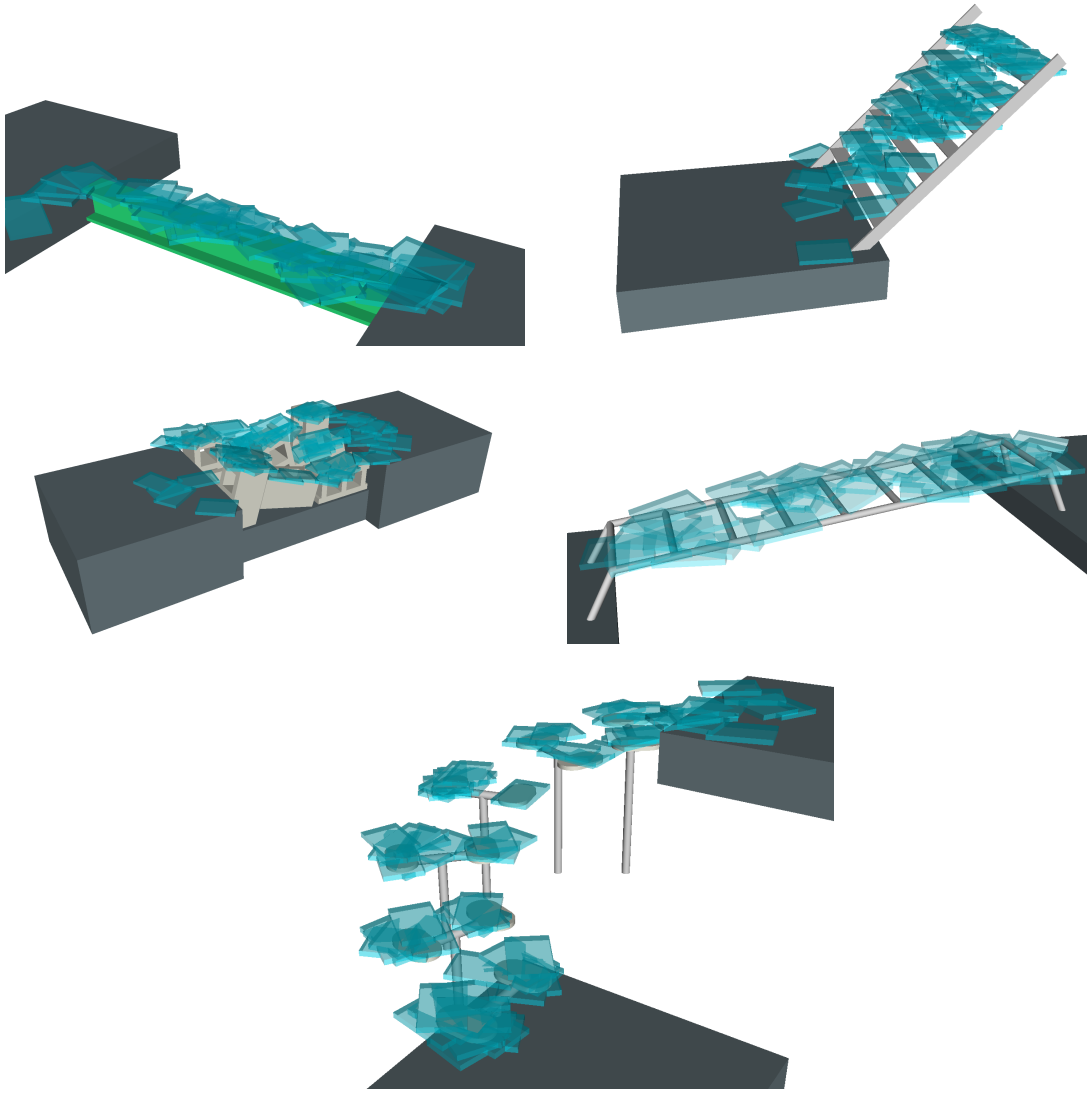
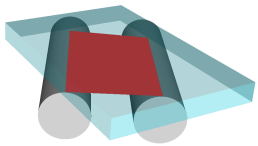
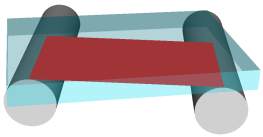


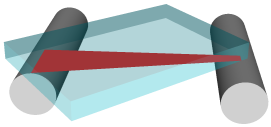
Figure 4.8: Examples of foot placement samples that were taken on a variety of environmental features.



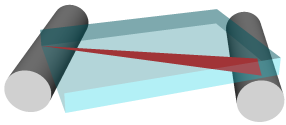
(a) 40% of foot length



(b) 80% of foot length



(c) 100% of foot length



(d) 110% of foot length

Figure 4.9: Two parallel bars can be used as a viable foot placement, depending on how far apart the bars are. As they get further apart, the availability of valid foot placements drops off.

Limitations

The key limitation of this sampling approach is that it relies on the ability to find a viable foot placement by performing rejection sampling on a plane. The method will struggle to identify pathological foot placements that have a low probability of being randomly sampled. Additionally, there may be geometric features which are capable of offering a viable foot placement in ways that are not tested for by this sampling method.

Low Probability Placements

Suppose two parallel bars have a distance between them that is close to the length of the foot. The foot would need to be placed somewhat precisely across the bars in order to constitute a viable foot placement. The probability of randomly sampling a precise placement is low, and shrinks rapidly as the necessary amount of precision grows. Figure 4.9 illustrates how the margin for viability shrinks at distances that exceed the foot length. Other

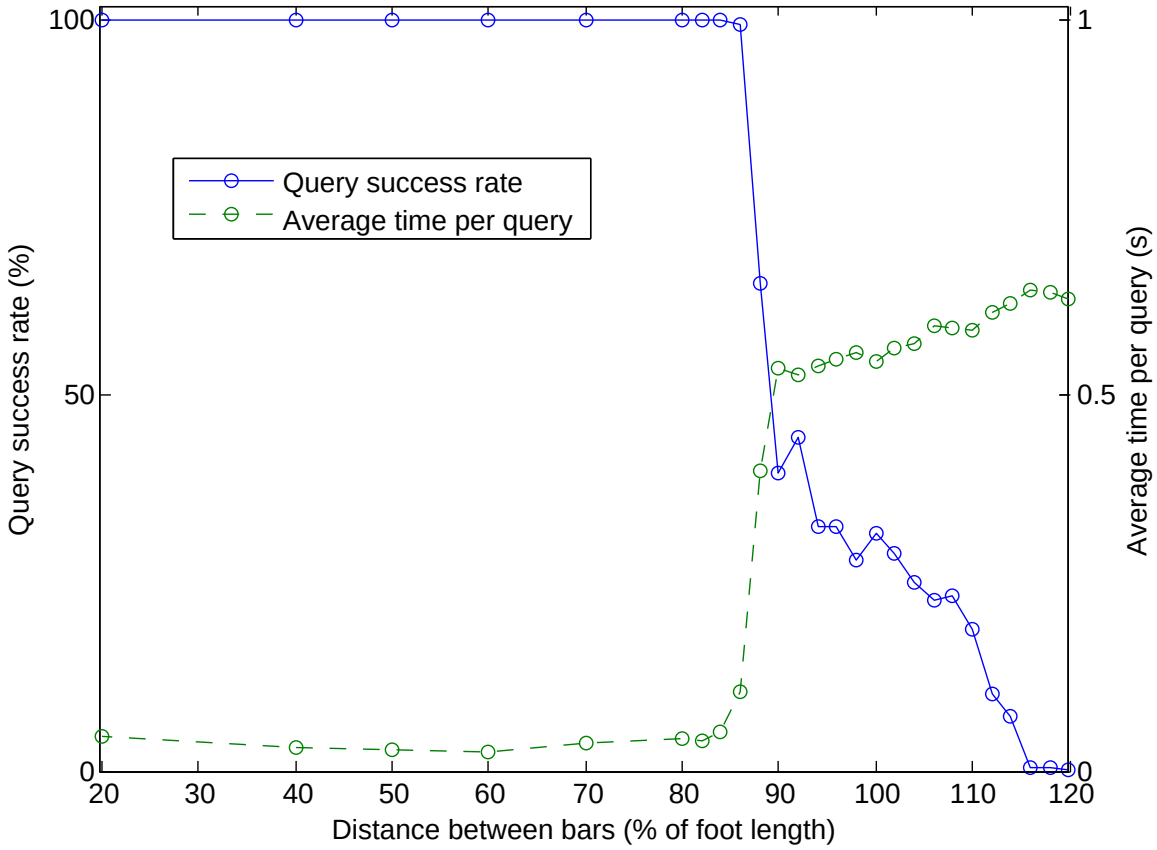


Figure 4.10: Experimental data for the parallel bar feature of Figure 4.9. The x-axis is the variation in distance between the bars, represented as a percentage of the foot length. For each data point, 1000 queries were made. The following parameters for Algorithm 1 were used: $N_{\text{SampleAttempts}} = 40$, $N_{\text{PlaneAttempts}} = 10$, $N_{\text{MaxIterations}} = 20$. A query is successful as soon as one viable foot placement is found and fails if all iterations are exhausted without one being found.

geometric features can also exhibit this behavior, although the probability of such features occurring in general is low, because a slight perturbation to the dimensions of the geometric feature could make the foot placement either much easier to find or no longer valid at all. Figure 4.10 shows experimental results for the parallel bar case.

Unrecognizable Placements

In addition to features with a low probability of being sampled, there are potential foot placements that have zero probability of being randomly sampled, or simply cannot be recognized by this sampling method. These foot placements tend to be in features that are

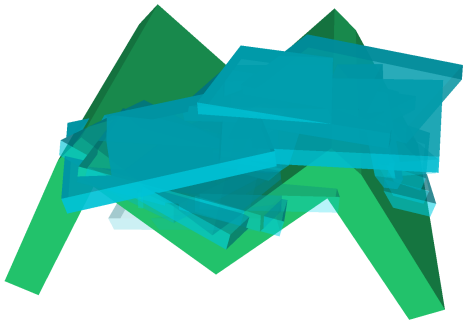


Figure 4.11: Concave-upwards features may be able to provide viable foot placements, but those placements cannot be recognized by the random sampling method.

concave-upwards where the topmost opening of the concave feature is wider than the sole of the robot's foot.

In the M-shaped feature of Figure 4.11, the robot could place its foot inside of the concave region with the "toes" (front edge) of the foot nested in the center corner of the M while the bottom of the foot lies flat against either of the interior faces of the M. However, our sampling method will not recognize it, because we assume that only the bottom of the foot will be in contact and that only friction on the bottom of the foot will be used to resist the component of gravity that is perpendicular to the surface normal. Since the interior faces of the M-shape exceed the maximum inclination, their planes are not examined for a viable foot placement—and even if they were, there would only be an infinitesimal probability of sampling a foot placement where the toes of the foot are perfectly touching the center corner of the M.

Elevation Projection

In 3D environments that exhibit significant elevation changes, the route exploration stage needs to be able to rise and fall with the terrain. The foot placement sampling method can be used to facilitate this. When attempting to expand $\Gamma_{\mathcal{P}}$ towards a target, if the current vertex fails to sample a viable foot placement, the vertex can be shifted up and/or down until a viable foot placement is identified, or until the vertex has exceed a maximum distance from

its parent vertex, at which point the extension procedure should terminate. This process is analogous to the projection operation used in CBiRRT [46] when moving a random sample onto the feasibility constraint manifold.

In some cases, it may be equally possible to project a vertex upwards or downwards. In such cases, the direction which would bring the vertex closer to its target should be preferred, since that is more likely to help to connect subtrees together.

When performing the projection operation, it is important to remember testing for collisions with the minimal geometry of the necessary conditions depicted in Figure 4.1a. Otherwise, the projection operation might accidentally allow the robot to phase down through a floor or up through a ceiling.

Whole Body Motion Planning

For the overall RPG to work, we also need to define $\pi^S_{\mathcal{M}}$ and $\pi^N_{\mathcal{M}}$ as described in Section 3.2. These should be motion planning methods which can take advantage of evaluations of the sufficient and necessary conditions, respectively. In this section, we will describe an implementation of a whole body motion planning method to suit $\pi^N_{\mathcal{M}}$. A planner that is suitable for $\pi^S_{\mathcal{M}}$ is then shown to be a special case of $\pi^N_{\mathcal{M}}$ which is applicable whenever the sufficient conditions C_S are satisfied.

Modes and Transitions

Multi-modal motion planning methods like MMPRM [25] and Random-MMP [39] use the concept of *modes* to define discrete changes in feasibility constraints. The feasibility constraint manifold of a mode σ is given by $\mathcal{F}_\sigma$, and represents the set of all configurations which satisfy the feasibility constraints of mode σ. In the work of Hauser, et. al. modes are determined by sets of contact points. A transition between modes occurs any time a new contact point is created or an old one is broken. When a system transitions from one mode, σ, to another, σ', its feasibility constraint manifold undergoes a discrete change from

$\mathcal{F}_\sigma$ to $\mathcal{F}_{\sigma'}$. The space of valid transition configurations between σ and σ', $T_\sigma^{\sigma'}$, is given by $\mathcal{F}_\sigma \cap \mathcal{F}_{\sigma'}$. If $T_\sigma^{\sigma'}$ is non-empty, then σ and σ' are considered to be adjacent modes.

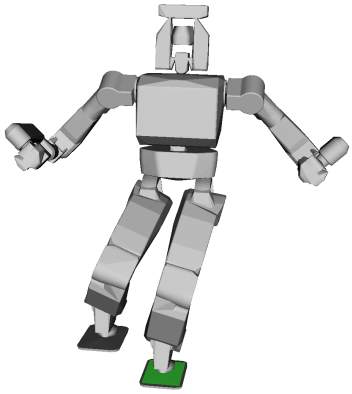
Since we are concerned specifically with flat-footed quasi-static bipedal locomotion, we can reduce the modes that are being considered into three categories: left-support, right-support, and double-support, based on whether the robot's weight is being supported purely by the left foot, supported purely by the right foot, or distributed between the two. A left-support or right-support mode can only transition to a double-support mode where the prior support foot remains in the same position. A double-support mode can only transition to a left- or right-support mode where the new support foot remains in the same position. Figure 4.12 illustrates the different categories of modes, as well as the transitions between them.

The purpose of a multi-modal motion planner is to find a feasible joint trajectory—which can transition between discrete modes—that moves the robot system from a start configuration to a goal configuration. The feasibility constraints of each mode are described by the four conditions mentioned in Section 4.1. By decomposing the search into an exploration of modes and the transitions between them, we can generate whole body motion plans while accounting for discrete changes in the system's feasibility constraint manifolds.

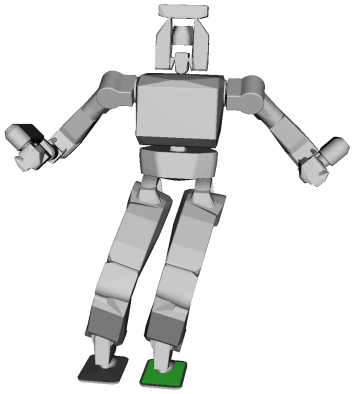
In our application, the multi-modal planner will be initialized with route information from stage $\mathcal{P}$ of the RPG. It will also contain a foot placement sampling phase which facilitates the creation of modes. A simple overview diagram of the process can be seen in Figure 4.13.

Transition Sampling

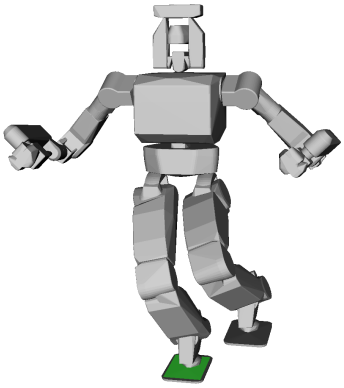
When handling a multi-modal motion planning problem, there may be many dozens, hundreds, or even thousands of modes to consider, but only a very small subset of them might actually be needed for a solution. Exploring all of the modes is likely to waste a consider-



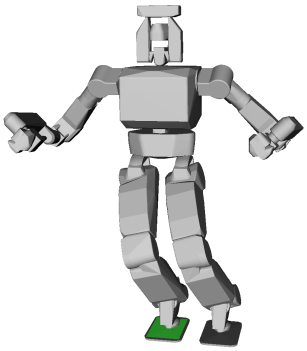
(a) A left-support mode. All weight is on the left foot, and the right foot is free to move unconstrained.



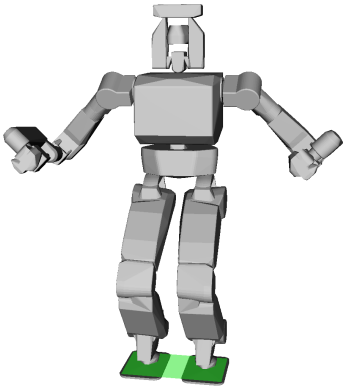
(b) A transition to or from a left-support mode. All weight is on the left foot, and the right foot is constrained to a specific location on the floor.



(c) A right-support mode. All weight is on the right foot, and the left foot is free to move unconstrained.



(d) A transition to or from a right-support mode. All weight is on the right foot, and the left foot is constrained to a specific location on the floor.



(e) A double-support mode. The weight is distributed between both feet, and both feet are constrained to specific locations on the floor.

Figure 4.12: Examples of the three categories of modes and transitions between them. The green polygon represents the support polygon of the current mode.

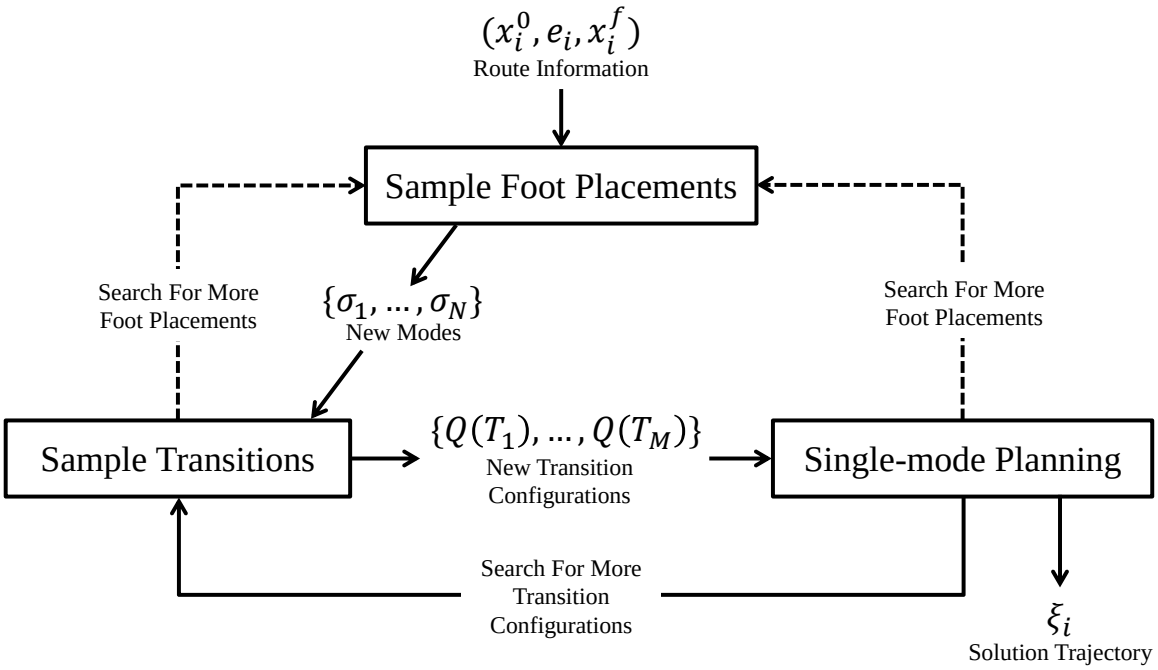


Figure 4.13: Workflow of the multi-modal motion planner. Dotted lines represent work paths that are only taken once there has been a large buildup in cost and more foot placements may be needed in order to find a solution.

able amount of time and effort. Instead, we recommend using strategies that maximize the expected value of the search effort.

In [38], Hauser and Latombe discuss a search strategy called *Search among feasible transitions* (SAFT). The observation made Hauser and Latombe is that transitions between modes are the most-constrained segments of a solution and therefore the least likely (and most difficult) to be found. If a feasible transition configuration exists between two modes σ and σ', then its existence lends credibility to the hypothesis that a feasible path exists from a configuration in σ to a configuration in σ'. A similar observation is made by Bretl et. al. in [51].

Since transitions between modes are the primary bottleneck of a multi-modal motion planner, we recommend focusing on identifying feasible transition configurations between modes, until there exists a sequence of such transitions that would be able to connect the start configuration to the goal configuration. After that, single-mode motion planning methods can be used to find feasible paths between transition configurations. If the planner

struggles to find the necessary single-mode paths, then more transition configurations can be sampled until a solution is found.

However, the number of transitions that may be available for consideration could scale as badly as $\mathcal{O}(M^2)$ where M is the current number of potential modes. Instead of uniformly sampling all transitions, we recommend first pruning transitions, and then focusing sampling effort on transitions that have a high likelihood of both existing and leading from the start to the goal.

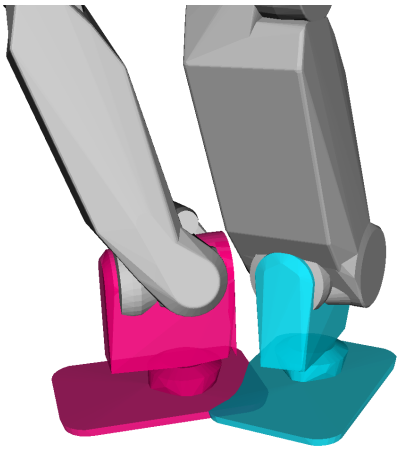
Pruning Transitions

There are a number of ways that infeasible transitions can be pruned to avoid wasting time on trying to sample them. First, any transitions which violate necessary conditions can be immediately pruned. The easiest necessary condition to test for is an outer approximation of the reachable space, $\mathcal{R}$. If the distance between the foot placements of the modes exceeds an upper bound on the radius of $\mathcal{R}$, then that transition can be immediately ruled out.

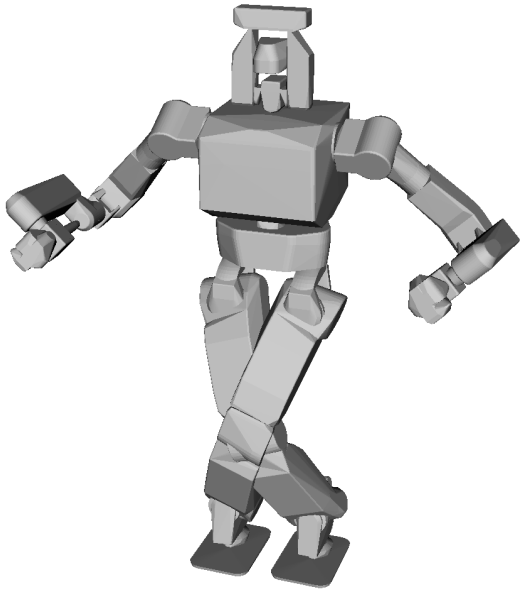
If the distance is within the upper bound, then inverse kinematics can be used to determined whether the foot placements can reach each other. If an analytical (a.k.a. closed-form) inverse kinematics solution is available for the robot's kinematic model, then we can definitively determine when the foot placements are not reachable. When only an iterative inverse kinematics solution is available, then a failed attempt to find a solution can be taken to indicate a low likelihood of a feasible configuration existing, but it cannot prune out the transition entirely. Instead of pruning the transition, we would simply assign it a very low priority when choosing which transitions to sample.

Two more examples of necessary conditions can be seen in Figure 4.14. If the robot's foot geometries would intersect when located at the modes' foot placements, then there is no way a feasible transition can exist. Similarly, if the foot placements would require the legs to intersect each other, then no feasible transition can exist.

Other methods also exist for pruning configurations. In [52] Hauser et. al. propose



(a) The foot placements for these modes force the feet to intersect, so this transition would be infeasible no matter what the robot's configuration is.



(b) While trying to satisfy the constraints of both modes, the legs cannot avoid intersecting each other.

Figure 4.14: Examples of cases where feasible transition configurations are guaranteed to not exist.

using a classifier for the feasibility of transitions. The classifier would be computed offline and would provide a quick estimate for the likelihood of a transition existing for any given pair of modes. This would also enable a *fuzzy* search [53] approach to the overall problem, by using sampling attempts to gradually adjust the estimated likelihood of hypothetical transitions that lead from the start to the goal.

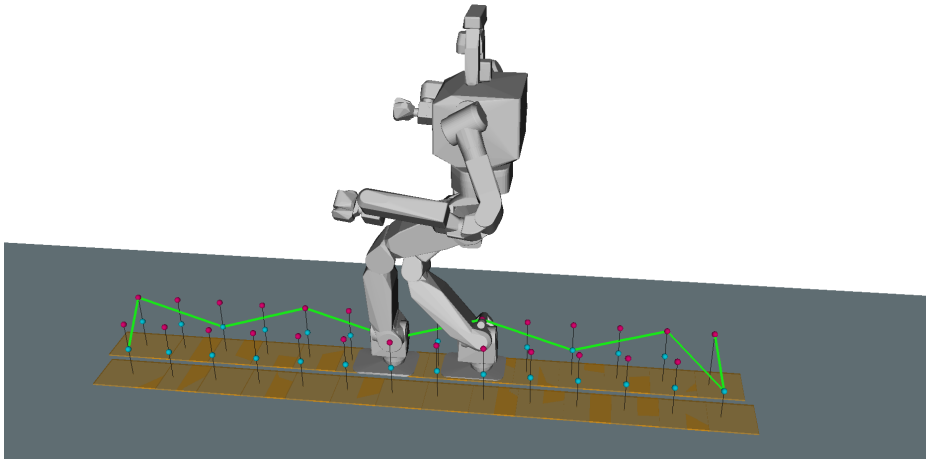
Prioritizing Transition Samples

Even after pruning infeasible transitions, there may still be an overwhelming number of transitions to consider. Here we describe a recommended approach for prioritizing which transitions get sampled. The goal of this prioritization is to maximize the expected likelihood that the transition sampling effort will lead to a solution.

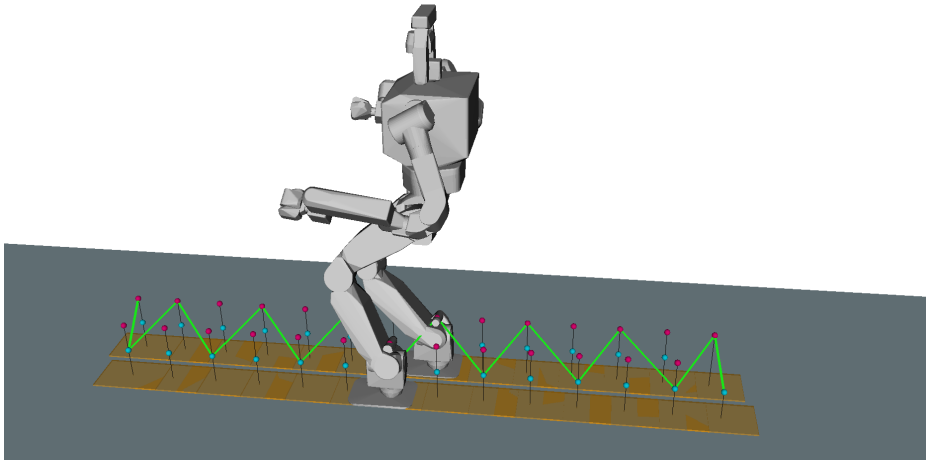
Hauser et. al. describe a priority queue of transitions [25, 38] where transitions at the top of the queue get sampled first, and the priority of each transition drops as sample attempts fail. A cost-based approach is described in [25] where the cost of a transition grows as attempts to sample it fail. This way, we can identify the lowest-cost transition sequence and focus the sampling effort along that sequence. This offers a holistic evaluation of the expected value of transition samples. In this section, we suggest some modifications to the costs described in [25]. These changes mostly relate to evaluating the cost of finding a solution rather than the evaluating what the quality of that solution would be once it is found.

First, we construct an adjacency graph where each mode is a vertex and each transition is an edge. The initial "cost" of each edge is measured by the cubed distance between the two foot placements belonging to the transition. The cubed distance is used instead of a simple linear distance because using the linear distance would encourage the robot to always take the largest possible steps, as illustrated in Figure 4.15. Taking the largest possible steps has several disadvantages:

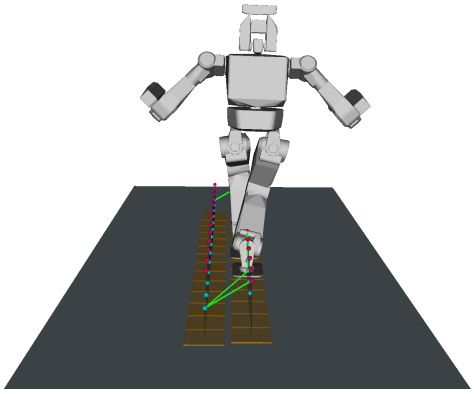
- Large steps tend to be inefficient since they require greater balance effort;



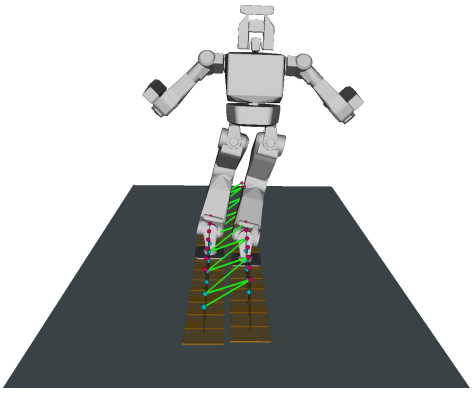
(a) Sideview of the lowest cost path of a straight walk when using linear distance for the edge cost.



(b) Sideview of the lowest cost path of a straight walk when using cubed distance for the edge cost.



(c) Backview of Figure 4.15a.



(d) Backview of Figure 4.15b.

Figure 4.15: Using the cubed distance to measure the cost of a transition encourages the planner to examine moderately sized steps before resorting to the largest possible steps. Blue (lower) dots represent the graph vertex of a left-support mode, and pink (upper) dots represent the graph vertex of a right-support mode. Orange boxes refer to the foot placement samples that are currently available to the planner.

- they are more difficult to control, because they force the robot to hug the edge of the feasibility constraint manifold;
- they are difficult to search for, because they exist very close to the space of infeasible configurations.

Each time a transition T_i is sampled, we increment a counter, $S(T_i)$, for that transition. When calculating the cost, we multiply the cubed distance between the feet by $S(T_i)$ in order to penalize transitions which have failed to find a feasible configuration. Note that $S(T_i)$ should be initialized to 1, even before any sample attempts have been made. Multiplying the edge cost by the number of failed sample attempts will force the planner to seek alternative paths whenever the edges along the preferred path are consistently failing to find feasible transition configurations.

Sometimes a feasible transition sequence may contain a small number of transitions which are particularly difficult to sample configurations for, but which will provide a feasible solution if queried enough times. To account for this, any edges which have successfully sampled a feasible configuration are assigned zero cost. This way, the planner can focus its efforts onto the small number of bottlenecks that may be preventing a solution from being found.

Once every transition in a sequence has successfully sampled at least one configuration, the single-mode planning can begin (described in Section 4.4.3). However, there may be cases where two adjacent transitions cannot be connected by a feasible path due to intermediate obstacles, or other constraints. Therefore, if the single-mode planning phase struggles to find valid paths between the transition configurations that have been found, it may be necessary to sample configurations along an alternative transition sequence. This motivates us to add a penalty, p, to transitions which are already being examined in the single-mode phase instead of assigning them zero cost.

Given a function $S(T)$ which returns the number of attempts that have been made to sample transition T, a function $D(T)$ which computes the distance between the foot place-

ments of T, a function $U(T)$ which returns a boolean which is true if T is being used in the single-mode planning phase (and false otherwise), and a function $Q(T)$ which returns the set of configurations that have been successfully sampled in T, we can define the cost of T, $C(T)$, with the following expression:

$$C(T) = \begin{cases} S(T)D^3(T) + p & \text{if } U(T), \\ S(T)D^3(T) & \text{if } \neg U(T) \wedge Q(T) = \varnothing, \\ 0 & \text{otherwise} \end{cases}$$

We recommend choosing the value of p to equal or exceed the highest cost of any transition T^* for which $U(T^*)$ is false. This helps ensure that the transition sampling phase will focus on alternative transition sequences instead of continuing to sample sequences which already have a set of feasible transition configurations that lead from the start to the goal.

Single-mode Planning

Once there exists a sequence of feasible transition configurations that lead from the start to the goal, the single-mode planning phase can be invoked. Planning within a single mode can be accomplished by any traditional motion planning method that can account for kinematic and geometric constraints. The Multi-modal PRM described in [25] uses a PRM-based planner called SBL [54].

Here we recommend using a variation of Constrained Bidirectional RRT (CBiRRT) [46], which grows trees bidirectionally from a set of start vertices and a set of goal vertices while remaining on a feasibility constraint manifold. Each transition configuration is a start or goal vertex.

We alternate the trees that we grow based on the expected likelihood of a solution being found that uses the tree. Similarly to how we select which transitions are sampled, we take a holistic approach to evaluating which trees to grow. We construct an adjacency graph

with modes as the vertices and transitions as the edges. Transitions which are empty (i.e. do not currently contain a valid configuration sample) are not given an edge in the graph. The path through the adjacency graph which has the lowest cost determines which modes are sampled. If no path exists, then this phase exits and returns to transition sampling.

The cost of each edge in the graph is a function of the transition associated with the edge and the modes that the transition connects. Costs should be assigned in a way that encourages the use of modes that are likely to assist in finding a solution while avoiding modes that are struggling to progress towards a solution.

The function $S(\sigma)$ keeps track of the number of attempts to extend trees within mode σ. The function $Z(T, \sigma)$ is a boolean function that returns true under the following conditions:

- There exists a configuration within transition T that is connected to a tree that leads to an overall start or overall goal configuration through mode σ, and
- the mode σ *only* contains vertices that connect to an overall start configuration or *only* contains vertices that connect to an overall goal configuration, not both.

If either of these conditions is violated, then $Z(T, \sigma)$ returns false. Illustrations of $Z(T, \sigma)$ can be seen in Figure 4.16 and Figure 4.17.

These functions can be used to define the cost that gets assigned to each edge of the adjacency graph in the single-mode planning phase. Consider transition T_i^j which is the transition that connects modes σ_i and σ_j. First we define a sub-cost function for each mode:

$$c(T_i^j, \sigma_i) = \begin{cases} 0 & \text{if } Z(T_i^j, \sigma_i), \\ S(\sigma_i) & \text{otherwise} \end{cases}$$

We can then use this sub-cost to define the full cost of the edge associated with T_i^j:

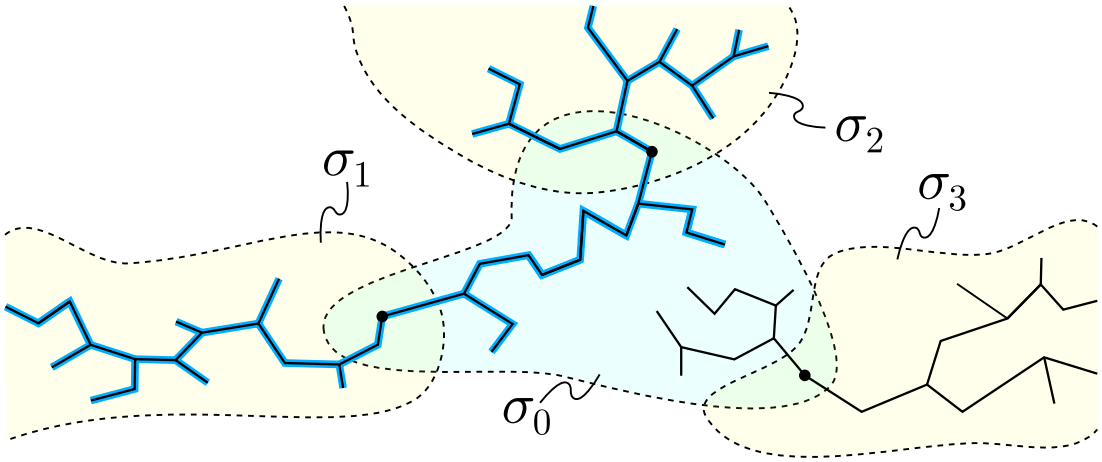
$$C(T_i^j) = D^3(T_i^j)[c(T_i^j, \sigma_i) + c(T_i^j, \sigma_j)]$$

Note that $T_i^j \equiv T_j^i$ because quasi-static motions are always reversible.

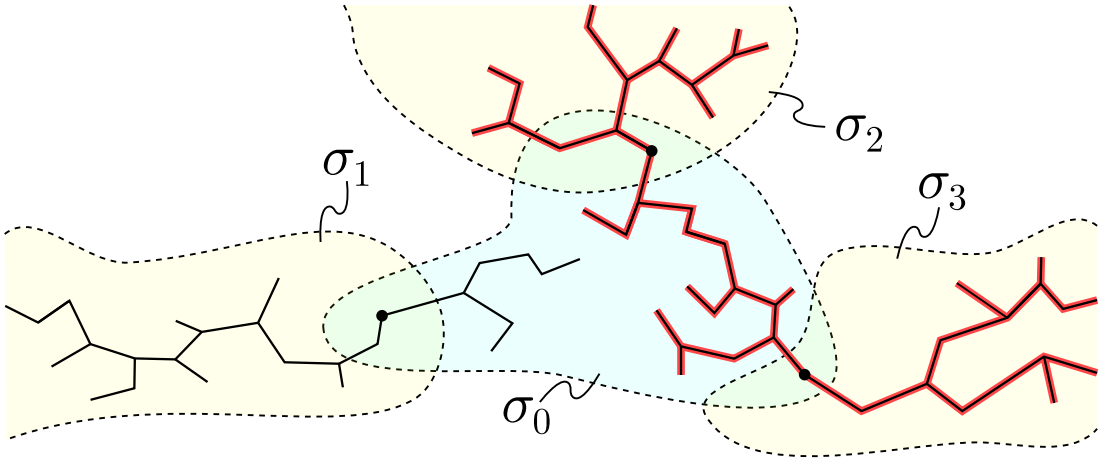
We can see in Figure 4.16a that $c(T_0^1, \sigma_0) = c(T_0^2, \sigma_0) = 0$. This is because a valid path through configuration space already passes through transitions T_0^1 and T_0^2 using mode σ_0, so it would cost us no further search effort to find a solution that passes through those transitions. The same argument can be made in Figure 4.16b for T_0^2 and T_0^3. Setting the costs of those transitions to zero incentivizes the single-mode planner to choose modes that will bridge the gap between the already existing start and goal trees.

However, there is a pitfall that we avoid by enforcing the second condition of $Z(T, \sigma)$. In Figure 4.16c without the second condition, we would have a situation where a zero-cost mode sequence would exist, even if there is no way for the start and goal trees to connect to each other within σ_0. Since it is impossible for any other path to have a lower cost than zero, the adjacency graph would always choose to search σ_0 for a connection between the start and goal—ignoring all other alternatives—even if σ_0 does not contain a viable path between the start and goal trees. The second condition of $Z(T, \sigma)$ prevents this from happening. Initially, σ_0 would be favored for a search because the mode sequences leading up to and away from σ_0 would have zero cost, but every time a search is attempted in σ_0, the value of $S(\sigma_0)$ would rise. Eventually the cost associated with σ_0 would be high enough that the adjacency graph would choose an alternative mode sequence.

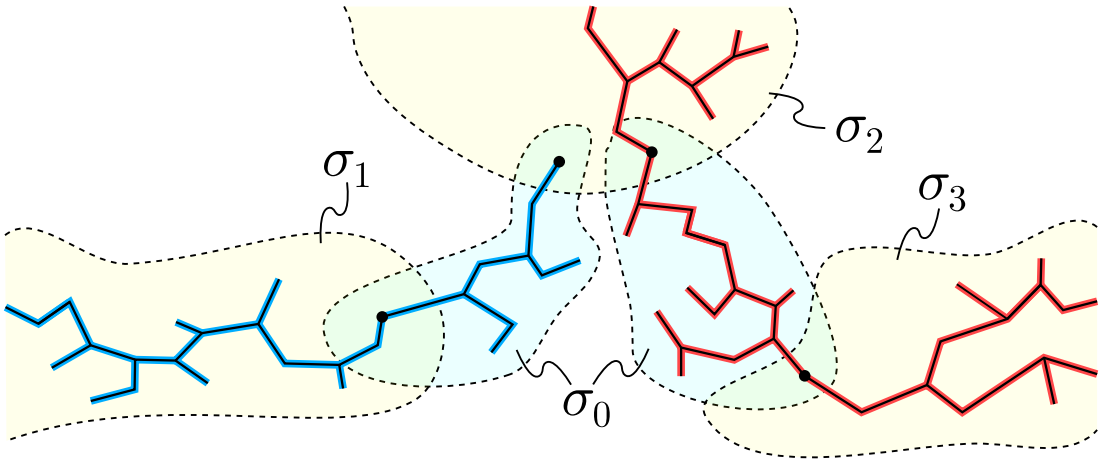
Once the lowest-cost sequence of modes is selected, we use CBiRRT within the selected modes. The trees that get used by the CBiRRT query are those which are growing from the previous and next transitions of each mode. These trees persist between each cycle of the planner. After every mode in the sequence has been searched, the planner returns to the transition sampling phase. Once a single tree exists that contains both the start configuration and the goal configuration, a complete solution has been found, and the low-level whole body motion planner has succeeded.



(a) $Z(T_0^1, \sigma_0) = Z(T_0^2, \sigma_0) = \text{true} \qquad Z(T_0^3, \sigma_0) = \text{false}$

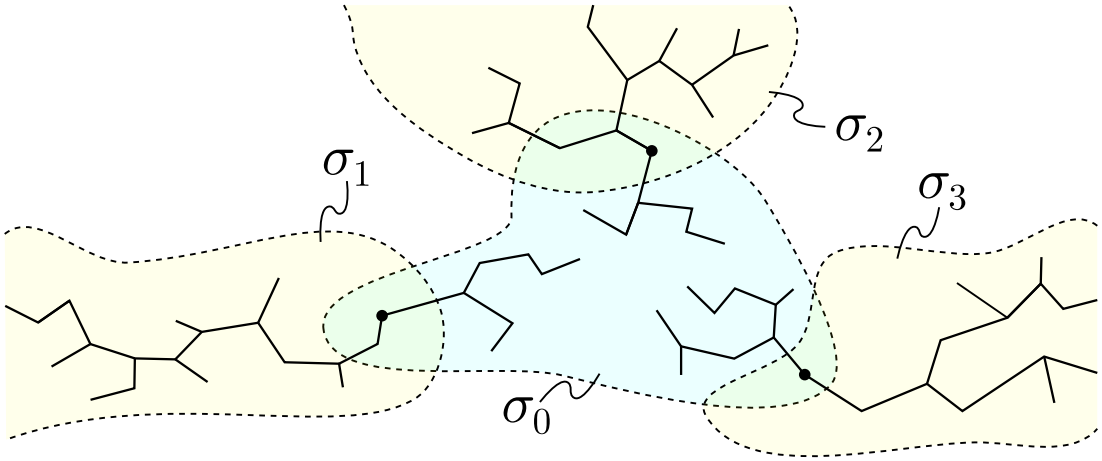


(b) $Z(T_0^1, \sigma_0) = \text{false} \qquad Z(T_0^2, \sigma_0) = Z(T_0^3, \sigma_3) = \text{true}$

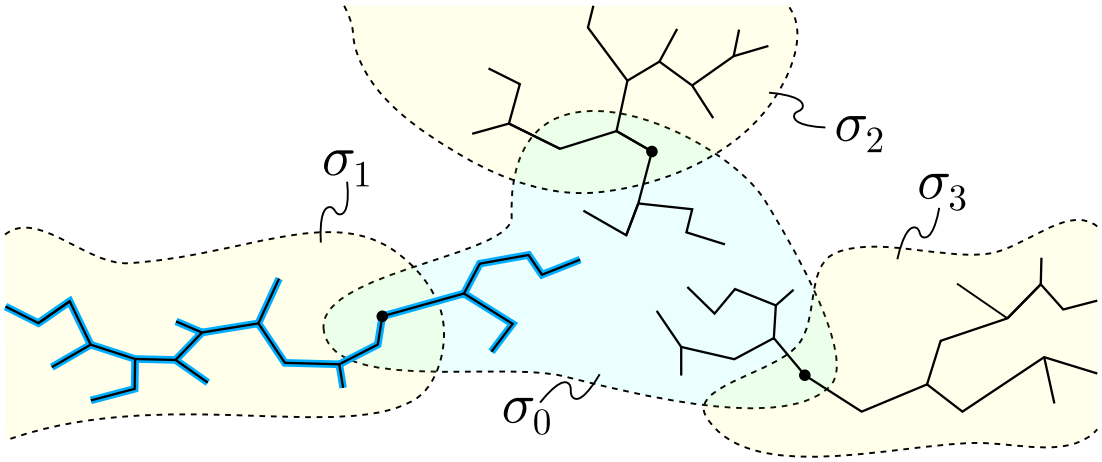


(c) $Z(T_0^1, \sigma_0) = Z(T_0^2, \sigma_0) = Z(T_0^3, \sigma_0) = \text{false}$

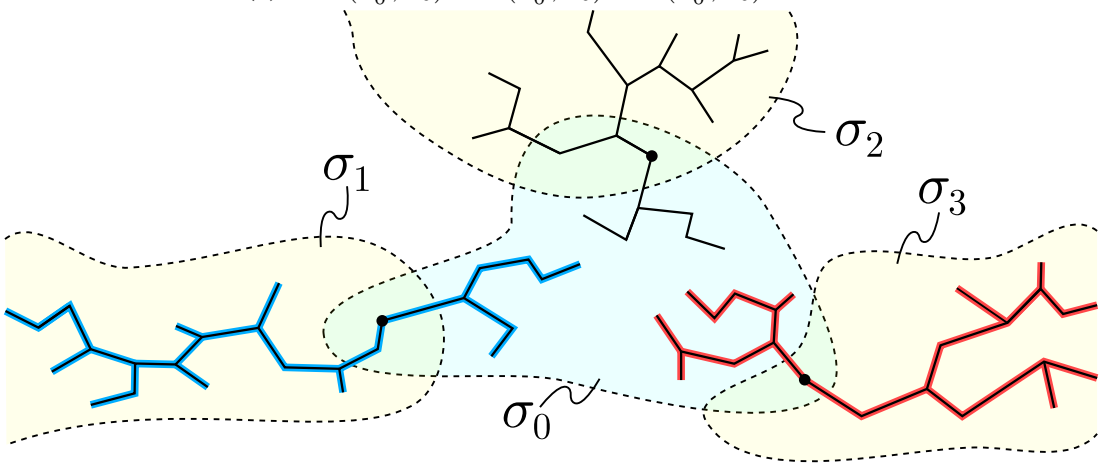
Figure 4.16: Blobs represent the feasibility constraint manifolds of modes. Black dots represent transition configurations, and lines represent the trees that have grown from them. Trees that are connected to an overall start configuration are colored blue, and trees that are connected to an overall goal configuration are colored red; otherwise they are black.



(a) $Z(T_0^1, \sigma_0) = Z(T_0^2, \sigma_0) = Z(T_0^3, \sigma_0) = \text{false}$



(b) $Z(T_0^1, \sigma_0) = Z(T_0^2, \sigma_0) = Z(T_0^3, \sigma_0) = \text{false}$



(c) $Z(T_0^1, \sigma_0) = Z(T_0^2, \sigma_0) = Z(T_0^3, \sigma_0) = \text{false}$

Figure 4.17: Continuation of Figure 4.16 showing more examples of how $Z(T, \sigma)$ is evaluated.

Incremental Footstep Sampling

Before transitions can be sampled or modes can be searched, a set of viable foot placements needs to be provided to the whole body motion planner. An initial set of foot placements can be found by sweeping the reachable space, $\mathcal{R}$, along the guide route provided by the high-level stage $\mathcal{P}$ of the RPG. If the foot placements that were sampled during stage $\mathcal{P}$ were cached, then those can also be recycled into the whole body motion planner.

Depending on the complexity of the terrain, the probability of successfully sampling a foot placement may be low in some regions. If the initial set of foot placements are not sufficient for the transition sampling stage to find a candidate transition sequence, then it will be necessary to immediately sample for more foot placements. Another sweep of foot placement samples along the route may be enough to fill in the missing gaps, but a more focused search may be more effective. For example, we can create two sets of foot placements: one which is reachable from the start configuration, and one which is reachable from the goal configuration. Then we can focus foot placement samples along the shortest route between the two sets of foot placements. If sampling along that shortest route fails to yield a candidate transition sequence, then the region of sampling can be expanded until a candidate sequence is found.

In some cases, it may be possible to find a candidate sequence of transitions to sample from, but they struggle to find a feasible sequence of transition configurations, which is necessary to begin the single-mode planning phase. When this happens, it may help to sample additional foot placements, which might yield transitions that can more easily find feasible configurations. These new samples can be focused near transitions which have a high $S(T)$ cost. When a transition T has a high $S(T)$ cost, it indicates that T is frequently being chosen as a high-value transition despite consistently failing to find a valid transition configuration. This usually corresponds to a bottleneck near the foot placements of T, and the bottleneck may be alleviated by providing additional foot placements in the region.

Similarly, the single-mode planning phase might also get stuck when confronted with

complex terrain. High values of $S(\sigma)$ can also be taken to indicate a bottleneck. Sampling foot placements near modes that have a high rate of sample attempts may provide easier alternative modes for the transition sampling and single-mode planning phases.

Each time more foot placements are added to the search, the complexity of the planning problem can scale up significantly. The transition sampling phase and single-mode planning phase both use a cost evaluation to focus the search effort along modes that are likely to provide a solution, but introducing new foot placements may produce many new modes that are initially evaluated as being low-cost. These new modes might distract the planner from continuing to search older modes that are close to finding a solution. Therefore, the introduction of new foot placements should be reserved for situations where the lowest-cost mode or transition sequences have risen beyond levels that are considered acceptable.

Primitives

Rather than relying only on randomly sampled configurations, we can use predetermined configurations and trajectories (which we will call "primitives") to improve the quality of motions that are found by the planner while also reducing the amount of search time required to find solutions. To maximize their utility, the primitives should be parameterized so that they can adapt to a range of situations.

Primitive Configurations

To assist in the transition sampling phase, it can be helpful to have a set of primitive configurations to seed the random search with. Whole body inverse kinematics (IK) solvers can adapt the primitive configurations to satisfy the constraints of the transition. If the set of primitives fails to find a solution, then samples can be randomized within a neighborhood of the primitive configurations. In the worst case, uniform samples of the joint space can be used, but uniformly random samples tend to be harder for an IK solver to adapt to the kinematic constraints of the transition than a set of high-quality primitives.

Primitive Trajectories

RRT-based methods are usually most effective in a wide open (expansive) free space. CBiRRT allows the RRT approach to work on low-dimensional (non-expansive) constraint manifolds, but at a cost: configurations need to be projected onto the constraint manifold using potentially costly inverse kinematics operations. Seeding CBiRRT's random sampling with configurations along a set of high-quality primitive trajectories can improve the likelihood of CBiRRT finding viable motions while also reducing the amount of IK effort required.

Chestnutt et al [37] use a set of adaptive parameterized primitive trajectories to define a discrete set of actions that can be utilized by a traditional search algorithm. The use of primitive trajectories in a randomized sampling method is described by Hauser et al in [25].

We can define the sufficient collision geometry of Figure 4.1b to fully encapsulate the robot's actual collision geometry while moving through all variations of the primitive trajectories. We can then identify the range of foot placements pairs which are reachable by any variation of the primitive trajectories as $\mathcal{R}_S$. If a pair of foot placements lies within $\mathcal{R}_S$, allows the sufficient collision geometry to not intersect the environment, and provides an adequate contact support base, then we can guarantee that a primitive trajectory will find a feasible motion using those foot placements. Fundamentally, this is what allows the performance gains provided by the sufficient condition checking of the RPG.

Post-Processing

The quality of an output trajectory can be evaluated based on various criteria, such as the efficiency, stability, and "naturalness" of the motions. The quality of output that gets returned by randomized samplers tends to be relatively low compared to most other locomotion planning methods, since randomized samplers are primarily concerned with finding *any* feasible plan within arbitrarily complex environments, rather than finding a high-quality plan. There are several ways to improve low-quality plans after they have been returned.

Shortcutting

A common approach to reducing random noise and improving efficiency in the output of a randomized planner is to find straight-line shortcuts between pairs of output configurations. This is not as straight-forward to apply on multi-modal plans, but it can offer considerable improvements.

Since the single-mode planning phase is greedy in the way it looks for a solution, it gives little consideration to the potential redundancy of the foot steps that it might be taking. This could cause it to step backward and forward repeatedly or to needlessly veer off to the side. Given an output trajectory with a sequence of modes which are known to lead to a solution, it may be possible to eliminate some of the foot steps that are used by the solution. We can essentially redo the transition sampling phase, but now we limit our search to only use the modes that were returned by the planner's output. This time we can focus the transition sampling to reduce the number of steps instead of maximizing the likelihood of finding a solution, since we already know that a solution can be found. The number of modes and transitions that need to be considered is likely to be orders of magnitude smaller than the original problem, making this search much less costly. When new transitions are found, we can attempt to connect those transition configurations to the trajectory of the original solution using CBiRRT.

After attempting to eliminate some of the foot steps, there may still be a considerable amount of random noise in the trajectory itself. This can also be reduced by shortcutting:

1. Randomly select a waypoint in the trajectory;
2. Find the set of all other trajectory waypoints which share its mode;
3. Randomly select a waypoint from the set that is not adjacent to the first waypoint;
4. Attempt a "straight-line" (or geodesic) connection between the two waypoints.

This procedure can be repeated until some criteria is met, e.g. a certain number of attempts are made or a time limit is reached.

Optimization

After shortcutting has been performed, the trajectory may be suitable for local optimization. A number of trajectory optimization approaches have been applied to locomotion planning, although many of them are designed to find trajectories and foot placements simultaneously [48, 55] and may be too limited in scope to apply to the kinds of plans that we will generate.

In our case, we want optimization methods that are general enough to handle arbitrary foot placements and environment obstacles. A good candidate for our purposes is CHOMP [56, 57] which was designed with high-quality locomotion planning via local optimization in mind.

Empirical Results

To test the performance and effectiveness of the RPG applied to bipedal locomotion planning, we constructed several virtual environments with a variety of challenging geometric features. In particular, we wanted to simulate environments where a height map representation (commonly used by traditional footstep planners) would be inadequate to accurately capture the environment's geometry. We also do not prescribe surfaces for the robot to walk on; this is left entirely for the planner to determine for itself. For implementation simplicity, all features in the environments are assumed to be rigid and unbreakable with a single coefficient of friction. Results were gathered on an Intel® Xeon® Processor E3-1290 v2 (8M Cache, 3.70 GHz) with 16GB of RAM.

In Figure 4.18 we present a simple scenario to test the planner's ability to find a plan that can mount and then immediately dismount a pair of beams that are obstructing the robot's path. Each beam is relatively thin, but they can provide a stable foothold if the foot is spread across both. This allows us to test the planner's ability to identify and utilize

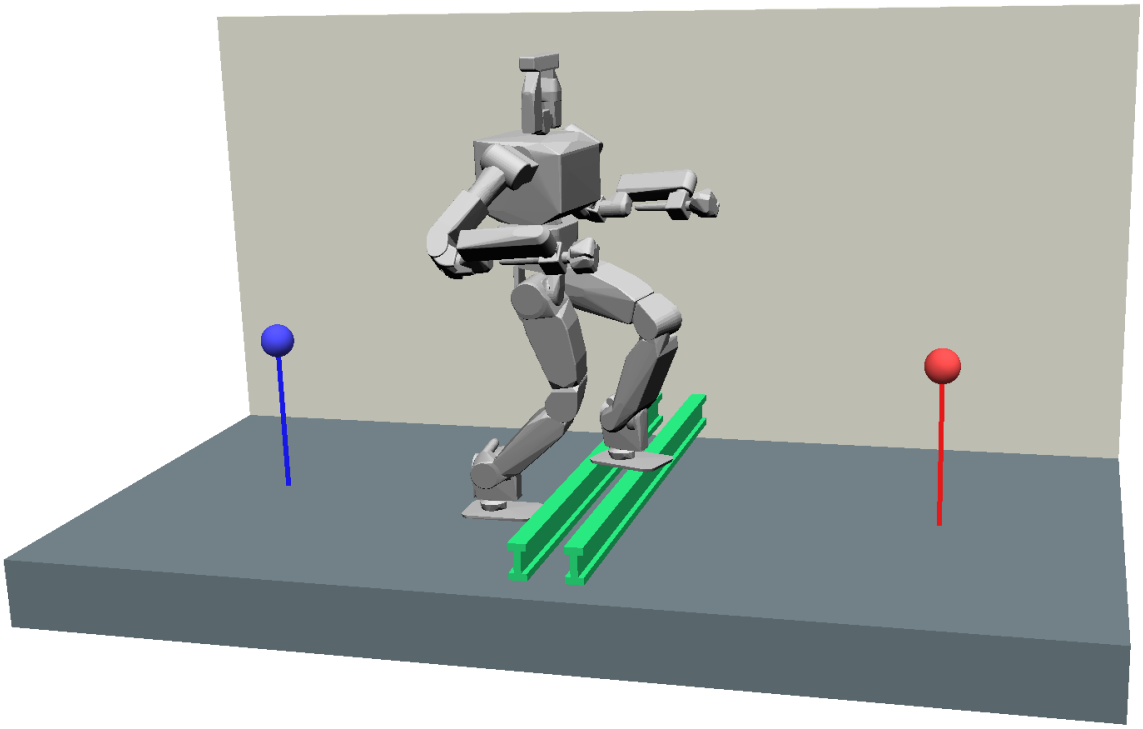


Figure 4.18: I-Beam Hallway. The robot is tasked with getting from one side of a hallway to the other. In the middle is a pair of I-beams that it needs to climb over. The blue and red markers represent the start and goal (respectively).

unorthodox foot placements.

Figure 4.19 is another simple scenario, this time designed to test the robot's ability to navigate across unstructured terrain with sharp changes in elevation and inclination. The center of the environment is littered with cinder blocks at various angles, which makes it difficult for the planner to find a direct path across them. The cinder blocks are assumed to be rigidly fixed in place and capable of supporting the robot's weight.

A more holistic test scenario is presented in Figure 4.20. We call this the "Jungle Gym" scenario, since it is an obstacle course which is loosely based off of features that may be found in a jungle gym. The robot's objective is to traverse from the southeast platform to the northwest platform. In the full version of the scenario, there are two viable routes that the robot may choose from. To see how the absence of this choice affects the performance results, we also test a version without the bars to walk across (Figure 4.20b) and a version

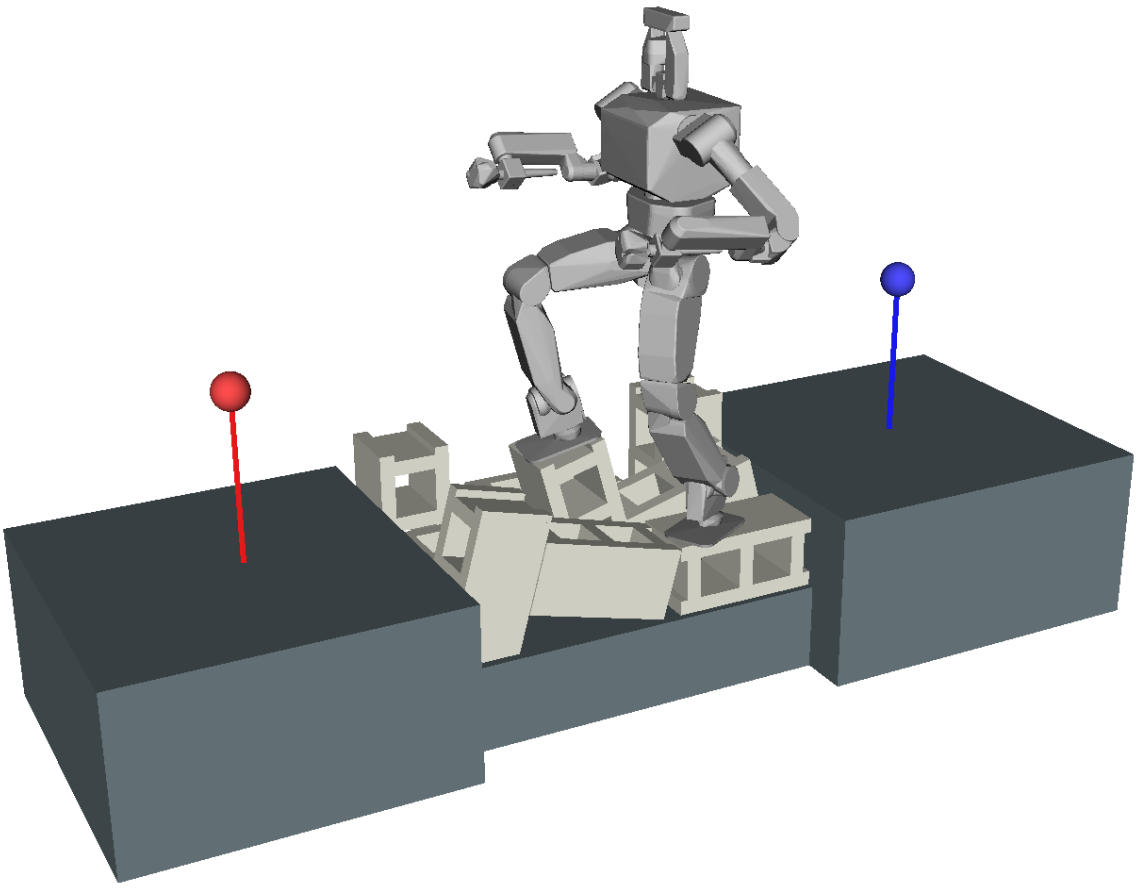


Figure 4.19: Cinder Blocks. The robot is tasked with getting from one side of a passage to the other. In the middle of the passage is a pit filled with tightly packed cinder blocks. The blue and red markers represent the start and goal (respectively).

Table 4.1: High-level performance results for the five scenarios using 100 trials each. Time measurements are given in seconds. The tests used a 1-hour timeout, after which a trial is considered a failure.

Scenario	**Avg Time (Std Dev)**	**Success Rate**
I-Beam Hallway (Figure 4.18)	30.59 (64.91)	100%
Cinder Blocks (Figure 4.19)	148.74 (375.90)	99%
Full Jungle Gym (Figure 4.20a)	1444.89 (813.77)	95%
Spiral Route (Figure 4.20b)	992.70 (685.02)	96%
Stair Route (Figure 4.20c)	2031.50 (917.95)	89%

Table 4.2: Performance results for each component of the Jungle Gym Scenario when they are tested independently. Each result is based off of 100 trials. A 1-hour timeout is used, after which a trial is considered a failure.

Component	**Avg Time (Std Dev)**	**Success Rate**
I-Beam	255.71 (135.84)	100%
Stairs	602.16 (498.21)	99%
Monkey Bars	803.69 (558.24)	97%
Spiral Steps	832.30 (677.08)	95%

without the I-beam (Figure 4.20c).

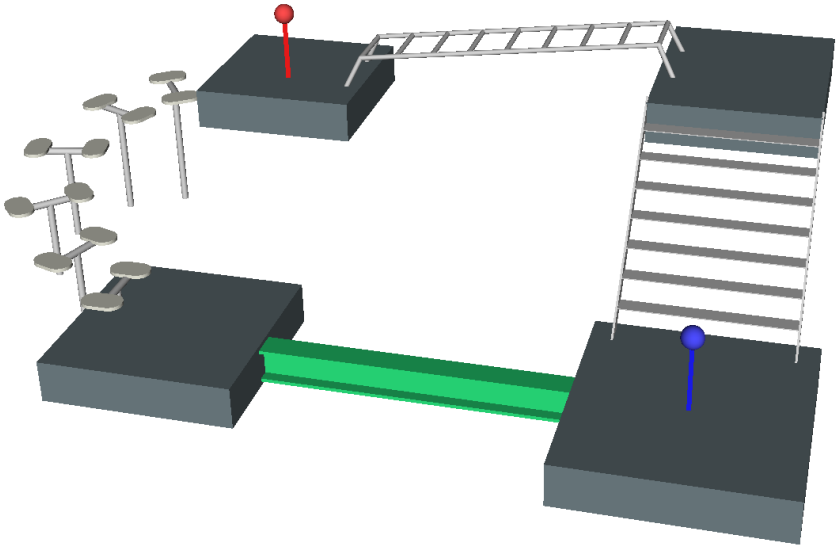
Table 4.1 lists the overall performance results for the five scenarios. Unsurprisingly, the I-Beam Hallway has the fastest performance since it is the simplest scenario. The Cinder Block Scenario is the second fastest; while it does have the most jagged terrain out of all the scenarios, it can be solved with a relatively small number of footsteps.

The Jungle Gym scenarios take significantly longer to solve than the two small-scale scenarios. This is due to a combination of the challenging nature of its terrain and the distance that the robot needs to travel from the start to the goal. Table 4.2 shows the breakdown of the time it takes to solve each environmental component individually, when dedicating all computational resources to each one.

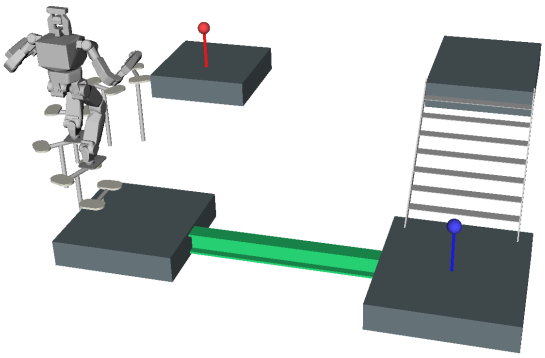
- The I-Beam is a relatively simple feature, but crossing it is challenging due to its narrowness: the robot needs to side-step or cross-step its way from one side to the other. It is not preprogrammed with primitives for either of those actions, so it must figure out how to perform them on the fly.

- The steepness of the stairs makes it impossible for the robot to walk up them forwards, because its shin would always collide with the next step. Instead, it must figure out how to turn its feet to the side and walk up at an angle. It may sometimes choose to face backwards while climbing up.

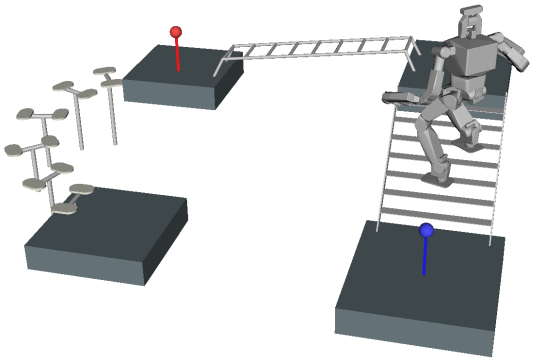
- The monkey bars are a challenging feature, since they require fairly precise foot



(a) Full Scenario

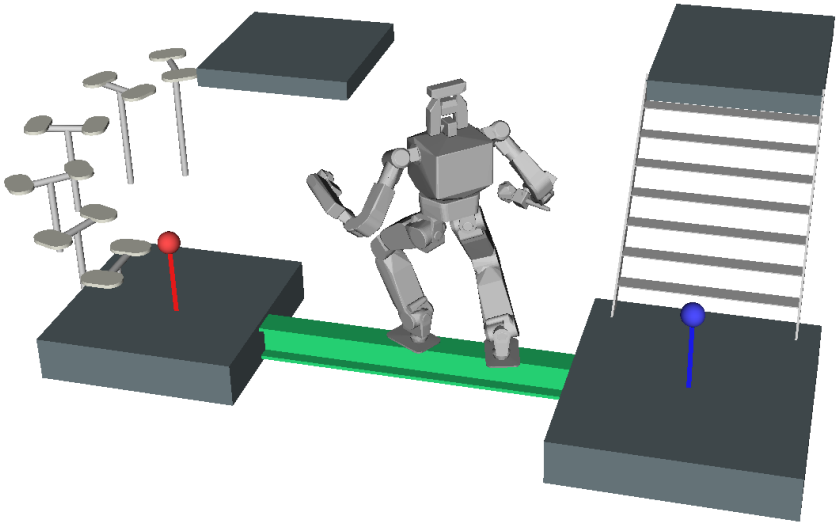


(b) Spiral Route: Monkey bars are removed. The only viable route is up the spiral steps.

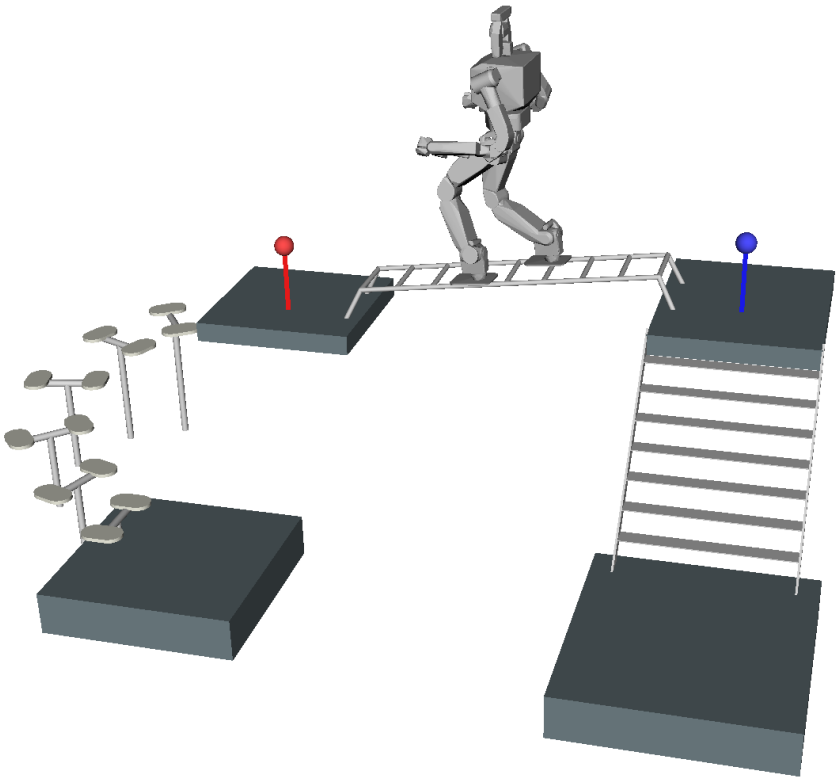


(c) Stair Route: I-beam is removed. The only viable route is up the stairs.

Figure 4.20: Variations of the Jungle Gym Scenario. The objective is for the robot to traverse from the blue marker in the southeast platform to the red marker in the northwest platform. In the full scenario, the robot may either (1) walk up the stairs on the east side and then walk across the monkey bars on the north side, or (2) walk across the I-beam on the south side and then up the spiral steps on the west side. We also test two variations on this environment to see how the robot performs when it no longer has a choice for which route to take.



(a) Crossing the I-Beam



(b) Crossing the Monkey Bars

Figure 4.21: Illustrations of the robot crossing two of the Jungle Gym components.

placements, and each foot placement has a relatively small support polygon.

- The spiral steps are much more open than the stairs, but they require more precise foot placements. They also require larger steps, which are inherently more difficult for the planner to solve.

The average times required by the Spiral Route (Figure 4.20b) and Stair Route (Figure 4.20c) are similar to the sums of the average times required to solve their components. This would suggest that the method should scale fairly well on increasingly complex scenarios, as long as the planner can find a way to decompose the scenarios using the necessary and sufficient conditions.

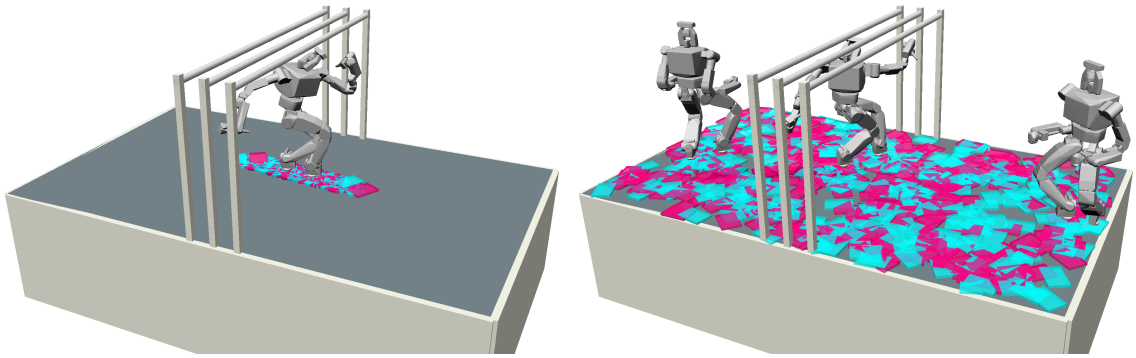
The time required by the full Jungle Gym Scenario is roughly the average of the time required by the two routes when they are split apart. This can be attributed to the fact that the planner's resources are being split evenly between investigating these two different options, since it does not know a priori which one will be easier to solve. Despite its resources being split between the two, the time that it takes is an average between the two instead of a sum of the two, because it is able to evaluate them in parallel and terminate once the easier one is solved.

CHAPTER 5
PROBABILISTIC COMPLETENESS FOR SEMI-UNSTRUCTURED ENVIRONMENTS

In this section, we examine the probabilistic completeness properties of the RPG method when applied to "semi-unstructured" environments. We define "semi-unstructured" to mean an environment that contains obstacles with arbitrary geometry, but where the walkable ground is flat and even. The restriction to flat and even ground is to ensure that all possible foot placements can be uniformly sampled, which allows the foot placement stage to be proven probabilistically complete. Unlike the procedure described in Section 4.2, we sample footstep locations on the flat ground in a way similar to the Task Space Region (TSR) method [58]. In the previous section, we discussed applying the RPG to arbitrary environments, but for that we used a foot placement sampling method which cannot guarantee uniform sampling—a necessary condition for the proof in this section. In the future, a more refined foot placement sampling method may be developed which can guarantee uniform sampling. For now, we limit the scope of our completeness analysis to flat ground so that we can directly utilize the TSR sampling method which allows for uniform random sampling.

In the theoretical examination, we consider a simplified algorithm which we will call the Worst-case RPG (w-RPG). The w-RPG exhibits the worst-case behavior of the ordinary RPG algorithm, which is what the ordinary algorithm would degenerate into when none of its built-in performance optimizations are effective. Since the ordinary RPG will have strictly better performance, the analysis of w-RPG represents a lower bound on the worst-case performance of the ordinary algorithm. Therefore, if the w-RPG is proven to be probabilistically complete, then the RPG is as well.

In addition to the proof of probabilistic completeness, we analyze a user-chosen param-



(a) $\rho = 0$ (b) $\rho = 5\mathcal{R}_{\text{max}}$

Figure 5.1: Illustration of the effect that the parameter ρ has on the Mode Sampling stage. The task for the robot is to pass underneath the set of three bars. Cyan and magenta boxes represent left and right (respectively) foot placement samples, and changing the value for ρ affects the size of the sampling region. $\mathcal{R}_{\text{max}}$ is the length of the largest step that the robot can take.

eter, ρ (see Figure 5.1), which affects the rate of convergence. This analysis provides hints for choosing a value for ρ that will provide reliable convergence. We also provide empirical data from simulation trials where the parameter is varied to demonstrate its quantitative impact.

Probabilistic Completeness

A process is considered probabilistically complete if the probability of it *failing* to find a solution when one exists converges asymptotically to zero as the number of samples it uses goes to infinity, i.e. the probability of failure can be written as:

$$\Pr[\text{FAILURE}] \leq \alpha \exp(-\beta N) \tag{5.1}$$

where α and β are positive constants greater than zero, and N is the number of samples being used by the process.

In pathological environments, α or β might be infinitesimal, in which case infinite samples would be needed to converge to a finite probability of finding a solution. Section 5.7.1 discusses the circumstances that lead to this for the w-RPG. In such cases, the w-RPG on

its own would not be adequate to find a solution.

Modes

Bipedal robots are hybrid dynamic systems (see Ames et al. [59, 60] for examples of detailed hybrid system models for bipeds) which exhibit sequences of discrete *modes*. In the scope of this section, a mode is defined by the placement of the support foot (or feet, in the case of double-support modes). Each mode corresponds to a set of feasibility constraints which determine whether a given configuration is physically viable for that mode. A mode takes the following form:

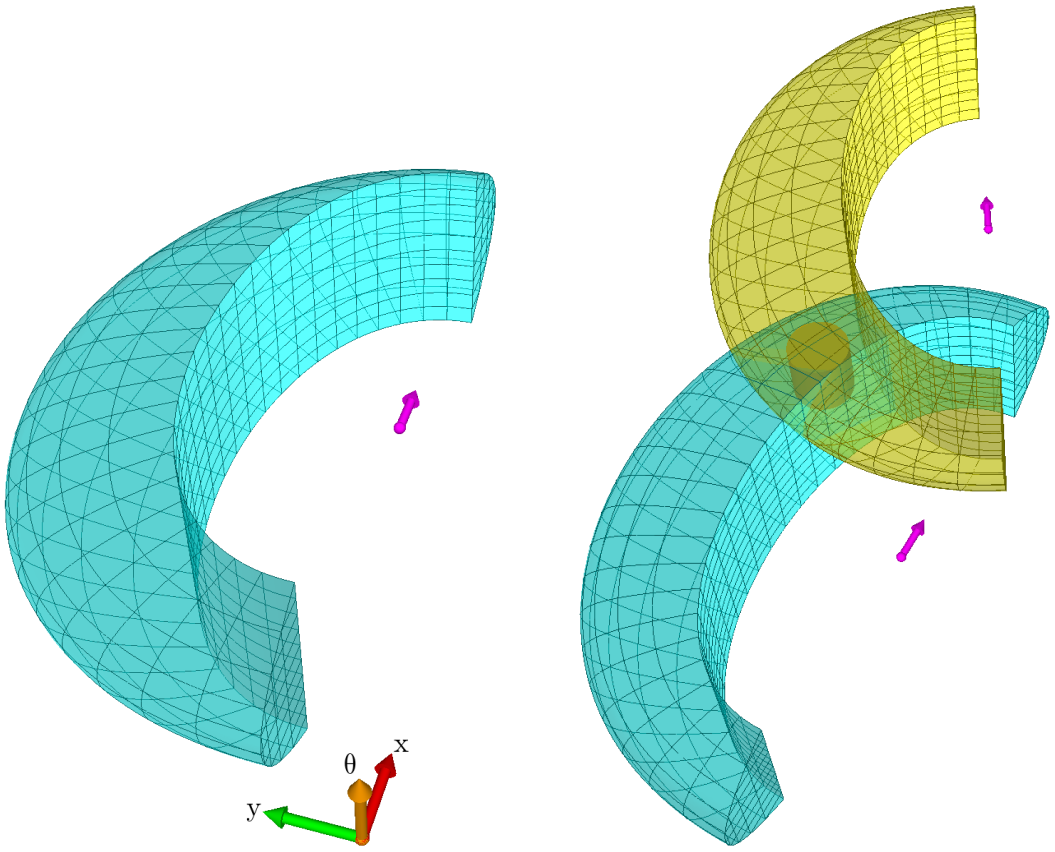
$$\sigma = \begin{cases} x_f \in \mathbb{R}^3 \mid f \in \{\text{Left}, \text{Right}\} & \text{if Single-Support} \\ (x_{\text{Left}} \in \mathbb{R}^3, x_{\text{Right}} \in \mathbb{R}^3) & \text{if Double-Support} \end{cases}$$

where x_f represents a foot placement consisting of two translational dimensions and one rotational dimension (for the yaw of the foot).

Worst-case Randomized Possibility Graph

For the analysis of this section, we consider a simplified version of the RPG scheme which we will call w-RPG. The simplified version discards the use of sufficient conditions and the performance benefits that come with them, so performance of w-RPG represents the worst-case performance of RPG. Using only the necessary conditions allows for more straightforward theoretical analysis and helps to establish an upper bound on the probability of failing to find a solution when one exists. There are three stages to w-RPG:

Possibility Exploration Instead of growing trees, we explore possibilities by sampling N_P points in the Possibility Exploration Space $\mathscr{E}$, which in this context is SE(3). Within the manifold of $\mathscr{E}$ is a submanifold called C_N which represents the points in $\mathscr{E}$ where the



(a) An example 3D representation of the space that can be reached by the left foot from a fixed location of the right foot (magenta arrow).

(b) An example of how the reachable spaces of two different foot locations can intersect. The interior cylinder is a subset of the left-foot locations that can be reached from both right-foot locations.

Figure 5.2: 3D illustrations of what a reachable space might look like for a bipedal system. Magenta arrows represent right-foot placements. Cyan and yellow regions are the corresponding reachable left-foot locations. The axes represent x/y translation and yaw.

simplified set of necessary conditions are satisfied. Any random samples which are not inside of C_N are rejected from the sample set. We then perform an $\mathcal{O}({N_P}^2)$ operation attempting to connect every pair of points with a "straight line". When a route through C_N is found that might be able to connect the start and goal states, this route is sent to the next stage: Mode Sampling.

Mode Sampling We sample modes uniformly near the route produced by the Possibility Exploration stage. The elements of the route are projected from SE(3) to $\mathbb{R}^2$, keeping only

the (x, y) values from the route. Then N_σ left and right foot placements are uniformly sampled within a radius ρ of each projected vertex along the route. The union of these circles is referred to as $\mathcal{F}_\sigma$. The orientations of the foot placements are uniformly sampled from $[0, 2\pi)$.

Once the foot placements are sampled, we perform an $\mathcal{O}({N_\sigma}^2)$ operation to test whether each pair of foot placements can reach each other. Each foot placement is assigned a single-support mode based on whether it is viable as a left- or right-foot placement. Each pair of foot placements that can reach each other are assigned a double-support mode.

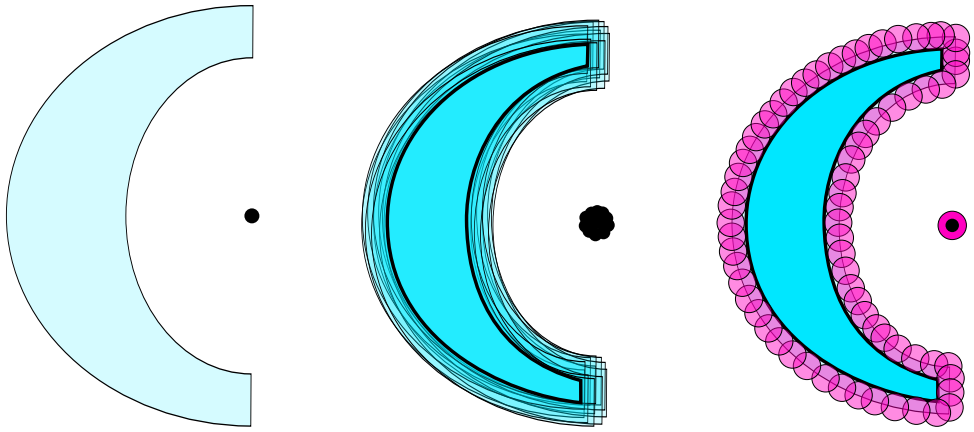
Multi-modal PRM Once a discrete set of modes have been sampled, Multi-modal PRM as described in [25] is used to find valid whole body paths through the modes.

Completeness of Mode Sampling

To have a viable sequence of modes, each mode in the sequence must be *adjacent* to the mode that comes before and after it. For two modes to be adjacent, their feasible spaces must intersect. A quick way to test for adjacency is to consider the kinematic reachability of one foot with respect to the other foot. For flat and even terrain, the reachable space is a function of the (x,y) position and yaw, θ, of the support foot. An illustration of what such a space might look like can be found in Fig. 5.2a. We assume that the reachable space is a subset of SE(2), containing at least one ball of radius $\epsilon > 0$.

For a sequence of modes to be valid, the foot placement of each single-support mode must be simultaneously reachable from the single-support modes that come before and after it, like the cylinder shown in Fig. 5.2b. The following lemma will help us show that there exists a region of foot placements wherein every placement is reachable from *every* member of a region of placements of the other foot.

Lemma 1. *Suppose we have a 2D shape, s (Fig. 5.3a). Consider the set of all possible translations of this shape within a circle of fixed radius r, $S = \{\sigma \in \mathrm{Trans}(s, \mathbf{x}) \mid |\mathbf{x}| < r\}$*



(a) (b) (c)

Figure 5.3: (a) A slice, s, of the reachable area for the left foot when the right foot is at the black dot. (b) Samples of the set S created by translating s around within a small radius. (c) The shape $\cap S$ created by the intersection of all elements within S. This is the same as the original shape, but contracted by circles around the border whose radii are equal to the maximum radius of the translations.

where $\mathrm{Trans}(s, \mathbf{x})$ *translates the shape* s *by vector* $\mathbf{x}$ *(Fig. 5.3b).*

Then the shape of the intersection of all elements in S*,* $\cap S$*, is equal to the shape of* s *contracted by circles of radius* r *densely packed around its border (Fig. 5.3c).*

Proof. Define b_s to be the boundary of s. The elements of s can be divided into two sets: $\alpha = \{x \in s \mid d(x, b_s) \geq r\}$ and $\beta = \{x \in s \mid d(x, b_s) < r\}$ where $d(x, b_s)$ computes the smallest distance between x and b_s.

An element $x \in s$ will **not** exist in the shape of $\cap S$ if and only if at least one shape in S was transformed by a distance greater than $d(x, b_s)$. Otherwise x cannot be outside the border of any shape in S.

By definition, the elements $x \in \alpha$ have the property $d(x, b_s) \geq r$, and every element of S was translated by less than r, so all of the elements of α must remain in $\cap S$.

Conversely, the elements $x \in \beta$ have the property $d(x, b_s) < r$. Since S contains elements which have been transformed by a distance up to r in every direction, the elements of β cannot remain in $\cap S$. Moreover, the elements of β are the same elements that would be covered by circles of radius r which are densely packed around b_s. An illustration of

this effect can be seen in Fig. 5.3. □

The effect of lemma 1 can be generalized to the 3D shape of Fig. 5.2a by continuously applying it to slices along the θ axis. If the original shape represented the space that is reachable from x_f, then the contracted shape would then represent the set of foot locations that can be reached from *any* location within a cylinder centered around x_f.

Now we can derive an upper bound on the probability of failing to sample a set of modes that can enable the system to reach the goal from the start, if such a set of modes exists. Assume there exists *some* solution, which is a function that outputs a configuration and a mode as a function of time:

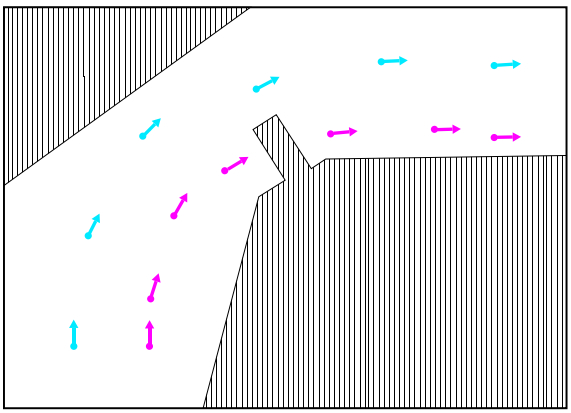
$$\gamma_S : [0, t_f] \longmapsto \mathbb{R}^{N_C} \times \Sigma$$

where N_C is the dimension of the configuration space and Σ is the set of all possible modes. The configuration output will vary continuously, but the sequence of modes through $[0, t_f]$ will be discrete and finite. Figure 5.4a displays an environment with the foot placements of a hypothetical solution that allows the robot to traverse from the bottom left to top right. We will now show that this selection of foot placements is not unique, and that uniform random sampling is a probabilistically complete way of finding a suitable sequence of modes to connect the start and the goal states.

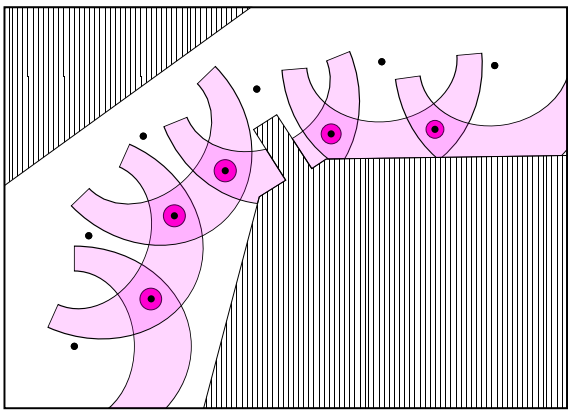
Theorem 1. *Let there be a sequence of M single-support modes $\{\sigma_1, ..., \sigma_M\}$ that are sufficient to connect a start state $x_{start} = (q_{start}, \sigma_{start})$ to a goal state $x_{goal} = (q_{goal}, \sigma_{goal})$. (Note that double-support modes exist between the single-support modes within the solution, but the double-support modes are not relevant to this theorem.)*

Then the probability that N_σ uniform samples of placements for each foot will fail to find a set of modes that can connect x_{start} to x_{goal} is at most

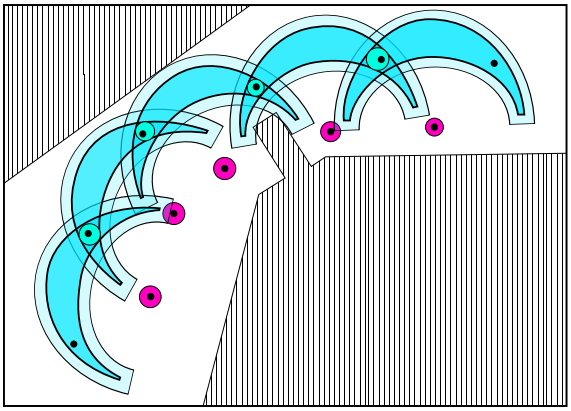
$$M\left(1 - \beta_m\right)^{N_\sigma} \tag{5.2}$$



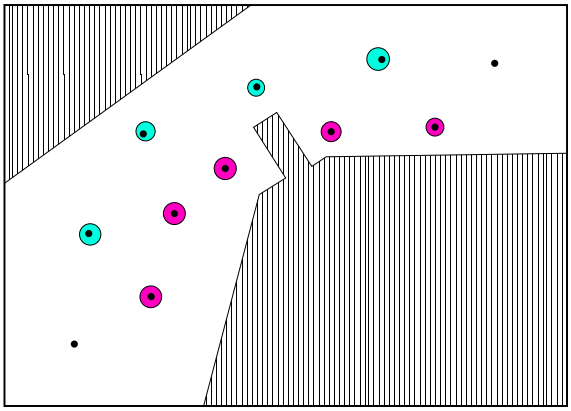
(a) Cyan and magenta arrows represent the foot placements of a hypothetical solution



(b) Magenta regions represent areas that the right foot can reach for each given left foot placement.



(c) Dark teal regions represent areas that the left foot can reach from any right foot placement within each magenta ball. The lighter teal border shows the original reachable shape, before being contracted.



(d) Each ball represents the foot placements that can be reached from any foot placement within the previous and next ball.

Figure 5.4: An environment consisting of regions where foot placements are valid (white) and invalid (striped). Foot placements may be invalid due to holes in the ground or obstacles on the ground.

where $0 < \beta_m \leq 1$ *is a problem-dependent constant.*

Proof. For a sequence of alternating single-support modes $\{\sigma_1, ..., \sigma_M\}$ to be valid, it is necessary for σ_{i+1} to be reachable from σ_i. Moreover, due to the symmetry of reachability, it is also necessary for σ_i to be reachable from σ_{i+1}.

The space of foot placements that are reachable from σ_i is given by the set $\mathcal{R}(\sigma_i)$. Therefore, for each mode $\sigma_{2i+1}, i = 0, ..., \lfloor \frac{M-1}{2} \rfloor$ we can identify a range of alternative foot placements by taking the intersection $\Sigma_{2i+1} = \mathcal{R}(\sigma_{2i}) \cap \mathcal{R}(\sigma_{2i+2})$. The foot placement for σ_{2i+1} can be replaced by any element in Σ_{2i+1} without affecting the validity of the mode sequence, because all elements in Σ_{2i+1} are reachable from the modes that come both before and after σ_{2i+1}. Examples of Σ_{2i+1} can be seen in the overlapping magenta regions of Fig. 5.4b.

Let us construct a cylinder named ς_{2i+1} of radius r_{2i+1} within each Σ_{2i+1} for $i = 0, ..., \lfloor \frac{M-1}{2} \rfloor$ (see Fig. 5.2b for a 3D illustration of such a cylinder, and Fig. 5.4b for an overhead view of a sequence of cylinders). For each ς_{2i+1}, the set of foot placements which are reachable by every member of the cylinder will be $\cap\mathcal{R}(\varsigma_{2i+1})$. From Lemma 1, we know that the shape of this intersection will be the ordinary shape of reachability but contracted by circles of $r(\varsigma_{2i+1})$ densely packed around the border. These contracted regions are illustrated in Fig. 5.4c. The cylinder also has a height, $\Delta\theta_{2i+1}$, which is chosen in conjuncture with r_{2i+1} such that the cylinder fits inside of Σ_{2i+1}.

Now for $i = 1, ..., \lceil \frac{M-1}{2} \rceil$ choose the largest cylinder available within the intersection $\{\cap\mathcal{R}(\varsigma_{2i-1})\} \cap \{\cap\mathcal{R}(\varsigma_{2i+1})\}\}$ and call it ς_{2i}. Note that σ_0 and σ_{M+1} are the start and goal (respectively) single-support modes which are given by the problem query. It is sufficient to have $\varsigma_0 \equiv \{\sigma_0\}$ and $\varsigma_{M+1} \equiv \{\sigma_{M+1}\}$, because both of those modes are provided without any sampling.

We now have a sequence of cylinders $\varsigma_i, i = 1, ..., M$ where as long as at least one foot placement from each cylinder is sampled, the set of samples will be sufficient for finding a valid solution that connects the start and goal states. Each cylinder is defined by its radius,

r_i and height, $\Delta\theta_i$. These parameters would ideally be chosen such that they maximize the volume of the smallest cylinder in the set. Choose r_m and $\Delta\theta_m$ to be the radius and height of the cylinder with minimal volume. The volume of this minimal cylinder is then $\pi r_m^2 \Delta\theta_m$.

Suppose we are given a planar region to sample from, $\mathcal{F}_\sigma$. Yaw values can simply be sampled from the range $[0, 2\pi]$. This gives us a sampling volume of $2\pi|\mathcal{F}_\sigma|$. If the x/y translations of the foot placements within each ς_i all lie in $\mathcal{F}_\sigma$, and we take N_σ independent samples of left-support modes and N_σ samples of right-support modes from $\mathcal{F}_\sigma$, then we get

$$\begin{aligned}
\Pr[\text{FAILURE}] &\leq \Pr[\text{Some cylinder } \varsigma_i \text{ is not sampled}] \\
&\leq \sum_{i=1}^{M} \Pr[\text{Cylinder } \varsigma_i \text{ is not sampled}] \\
&= \sum_{i=1}^{M} \left(1 - \frac{\pi r_i^2 \Delta\theta_i}{2\pi|\mathcal{F}_\sigma|}\right)^{N_\sigma} \\
&\leq M \left(1 - \frac{\pi r_m^2 \Delta\theta_m}{2\pi|\mathcal{F}_\sigma|}\right)^{N_\sigma}
\end{aligned}$$

which gives us

$$\Pr[\text{FAILURE}] \leq M \left(1 - \frac{r_m^2 \Delta\theta_m}{2|\mathcal{F}_\sigma|}\right)^{N_\sigma} \tag{5.3}$$

If we then take

$$\beta_m = \frac{r_m^2 \Delta\theta_m}{2|\mathcal{F}_\sigma|}$$

we know that $0 < \beta_m \leq 1$ because r_m and $\Delta\theta_m$ are non-zero (except in pathological cases), and the volume of the sampling space must be at least as large as the volume of the smallest cylinder in order to satisfy the assumption that $\mathcal{F}_\sigma$ covers all foot placements in each set ς_i. Therefore, substituting β_m into equation 5.3 gives us the expression in equation 5.2. □

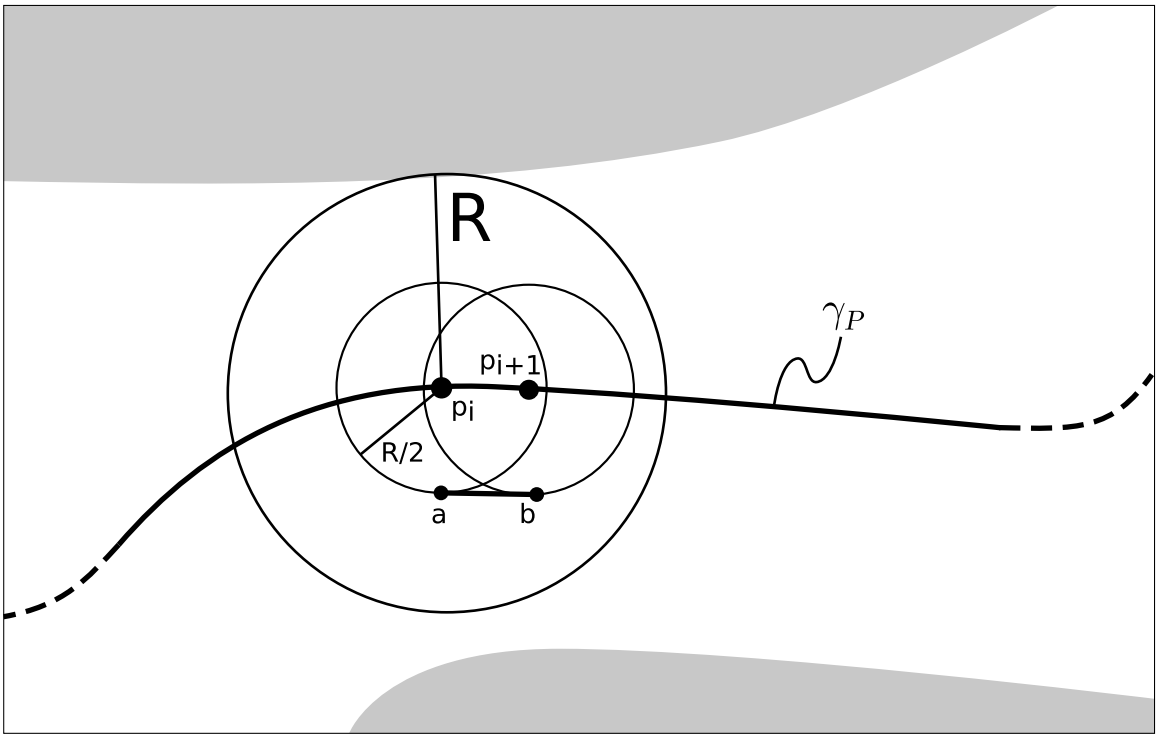


Figure 5.5: Illustration for the proof of Lemma 2. The white area represents C_N while gray is $\mathscr{E} \setminus C_N$. R is the minimum distance between the path γ_P and the edge of C_N.

Completeness of Possibility Exploration

Now we consider the Possibility Exploration stage, where we find samples that exist in the necessary condition manifold, C_N, and connect them in a graph using geodesics ([61] provides useful implementation details for sampling points in SE(3) and connecting them). We derive an upper bound for the probability that N_P samples will fail to provide a route that can be used by the Mode Sampling stage to find adequate mode samples for a solution. This proof is largely derived from the proofs of probabilistic completeness presented by [62] and [63], but we also account for the need to obtain an adequate sampling of foot placements, which was not a requirement for prior proofs.

As before, assume a solution exists in the form:

$$\gamma_S : [0, t_f] \longmapsto \mathbb{R}^{N_C} \times \Sigma$$

We can transform this function into

$$\gamma_P(s(t)) = \text{Proj}_{\mathscr{E}}(\gamma_S(t))$$

where $\text{Proj}_{\mathscr{E}}(x)$ is a function that projects a state x into the Possibility Exploration Space, and $s(t)$ parameterizes γ_P by arclength instead of time.

Definitions: We denote $d_{\gamma_P}(s, r)$ to compute the arclength distance between points $\gamma_P(s)$ and $\gamma_P(r)$ along the curve γ_P. We define $B_r(s)$ to be the set of all points in $\mathscr{E}$ within a ball of radius r centered at $\gamma_P(s)$. Recall that $\mathscr{E}$ is the "Exploration Space" from which we randomly sample points to see if they satisfy the necessary conditions. In the context of this paper, $\mathscr{E}$ is equal to SE(3) where the translational dimensions are bounded by a box. d_{γ_P} therefore computes a distance in SE(3). For more information on distance metrics in SE(3), see [61].

Lemma 2. *Let $\gamma_P : [0, L] \longrightarrow \mathscr{E}$ be a path that connects $p_{start} = Proj_{\mathscr{E}}(x_{start})$ and $p_{goal} = Proj_{\mathscr{E}}(x_{goal})$. Let $R = \inf_{0 \le s \le L} r(\gamma_P(s))$ be the minimum distance of the path to the edge of the necessary condition manifold C_N.*

Then the probability that N_P uniform samples of C_N will fail to yield a path that can connect from p_{start} to p_{goal} is no greater than

$$\frac{L}{\varepsilon}\left(1 - \frac{\pi^3 \varepsilon^6}{6|C_N|}\right)^{N_P} \tag{5.4}$$

where $0 < \varepsilon \le R/2$, and $|C_N|$ is the volume of the necessary condition manifold.

Proof. Let $n = \lceil L/\varepsilon \rceil$. We can then find a set of points $\{p_0 = p_{\text{start}}, p_1, ..., p_n = p_{\text{goal}} \in \gamma_P \mid \forall i, d_{\gamma_P}(p_i, p_{i+1}) \le \varepsilon\}$. Note that

$$B_{R/2}(p_{i+1}) \subseteq B_R(p_i), \text{for } i = 0, ..., n-1. \tag{5.5}$$

This follows from the triangle inequality and the inequality $|\gamma_P(s) - \gamma_P(r)| \le d_{\gamma_P}(s, r)$.

Assume we have the points $a \in B_\varepsilon(p_i)$ and $b \in B_\varepsilon(p_{i+1}))$. If we enforce $\varepsilon \leq R/2$, then $B_\varepsilon(p_i) \subseteq B_{R/2}(p_i)$, and equation 5.5 guarantees that both $a, b \in B_R(p_i)$. Therefore, there is guaranteed to be a geodesic line segment $\overline{ab}$ that lies entirely within C_N and connects the points a and b, because every point in $B_R(p_i)$ lies within C_N due to the definition of R. This property is illustrated in Fig. 5.5.

This observation tells us that it is sufficient to have at least one sample point in each ball $B_\varepsilon(p_i), i = 1, ..., n-1$ for the Possibility Exploration stage to find a path that connects the start point to the goal point, as long as $\varepsilon \leq R/2$. We can sample SE(3) from $\mathbb{R}^6$ without loss of generality using an Euler angle representation of orientation. Therefore the volume of the balls to be sampled can be computed based on a 6-ball: $\pi^3\varepsilon^6/6$. Taking N_P independent samples from C_N, we find

$$\begin{aligned} \Pr[\text{FAILURE}] &\leq \Pr[\text{Some ball is not sampled}] \\ &\leq \sum_{i=1}^{n-1} \Pr[\text{Ball } B_\varepsilon(p_i) \text{ is not sampled}] \\ &\leq \frac{L}{\varepsilon}\left(1 - \frac{\pi^3\varepsilon^6}{6|C_N|}\right)^{N_P} \end{aligned} \tag{5.6}$$

□

Definition: $d_{xy}(\sigma, p)$ computes the distance across the xy-plane between the foot location corresponding to the mode σ and the point p.

Lemma 3. *As in Lemma 2, $\gamma_P : [0, L] \longrightarrow \mathscr{E}$ is a path that connects p_{start} and p_{goal}. Let h_m represent the greatest distance of any foot placement in the union $\cup\varsigma_i, i = 1, ..., M$ from the point on the path γ_P which is closest to that mode:*

$$h_m = \sup_{\sigma \in \cup_i \varsigma_i} \inf_{s \in [0,L]} d_{xy}(\sigma, \gamma_P(s)) \tag{5.7}$$

Given a value of $\rho \geq 2h_m$ (see Fig. 5.6), the probability that N_P uniform samples of C_N

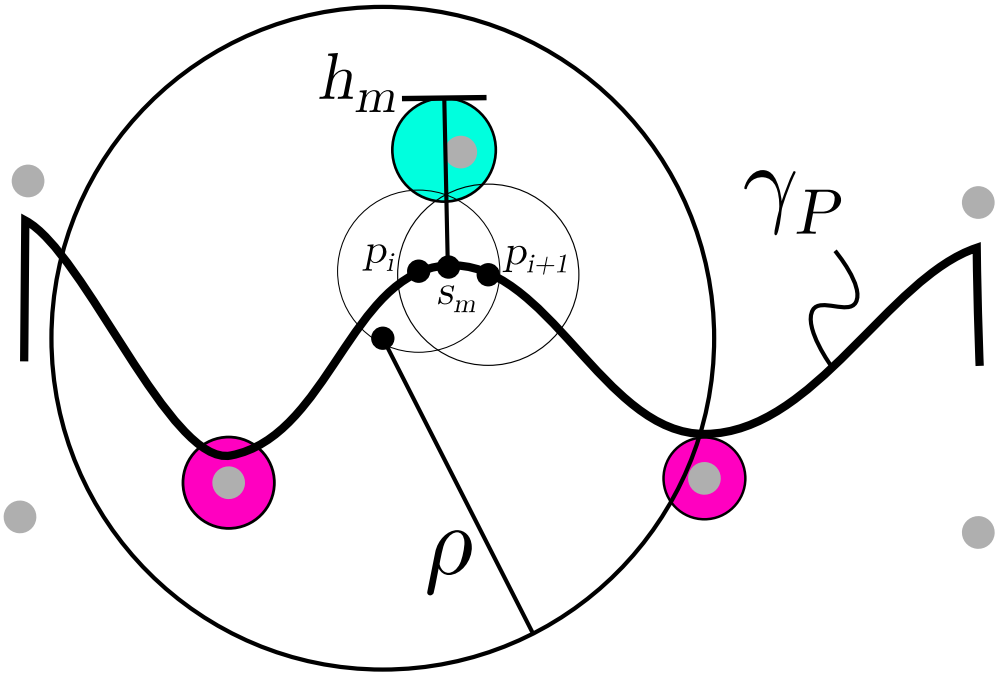


Figure 5.6: Illustration of the parameter h_m. Teal and magenta balls represent the cylindrical regions of acceptable foot placements, ς_i, from Sec. 5.3, and gray dots represent the foot placements that are used by the hypothetical solution of γ_S. Small black dots are points in the Possibility Exploration Space $\mathscr{E}$.

will not be adequate to sample the modes needed for a solution is no greater than

$$\frac{L}{\varepsilon}\left(1 - \frac{\pi^3\varepsilon^6}{6|C_N|}\right)^{N_P} \tag{5.8}$$

where $0 < \varepsilon \leq \rho/4$ *and* $|C_N|$ *is the volume of the necessary condition manifold.*

Proof. As in the proof for Lemma 2, let us define a set of $n = \lceil L/\varepsilon \rceil$ points $p_0 = p_{\text{start}}, p_1, ..., p_n = p_{\text{goal}}$ along γ_P such that $d_{\gamma_P}(p_i, p_{i+1}) \leq \varepsilon$ for each $i = 0, ..., n-1$.

Suppose we choose ρ such that $\rho \geq 2h_m$. This condition is easily enforced using known information by setting ρ to be at least double the furthest distance that the robot can step, which we will refer to as $\mathcal{R}_{\text{max}}$. Additionally, suppose we enforce $\varepsilon \leq \rho/4$. Define s_m to be the minimizer for s in equation 5.7. Define p_m to be the point from the set $\{p_0, ..., p_{n-1}\}$ that is closest to the value s_m. We know that p_m cannot be further than $\rho/4$ from s_m, or else another ball would have been placed in the sequence, and that new ball would be closer to

s_m than p_m, which would contradict the definition of p_m. Therefore, no point in $B_{\rho/4}(p_m)$ can be further from $\gamma_P(s_m)$ than $\rho/2$.

Define σ_m to be the maximizer for σ in equation 5.7. If x_m is the translational location of the foot placement for σ_m, then x_m has a distance h_m from $\gamma_P(s_m)$. Therefore, the triangle inequality tells us that the furthest distance that x_m could possibly have from p_m is $\delta_m \leq \rho/2 + h_m \leq \rho$. Since x_m is the furthest possible foot placement, all other foot placements in the union of $\varsigma_1, ..., \varsigma_M$ must be within a distance $\delta_j \leq \delta_m \leq \rho$ of every point within some ball $B_\varepsilon(p_i), i = 0, ..., n-1$, as long as $\varepsilon \leq \rho/4$. Figure 5.6 illustrates this property.

Therefore, as long as $\rho \geq 2h_m$ and $\varepsilon \leq \rho/4$, it is sufficient to have at least one sample point in each ball $B_\varepsilon(p_i), i = 1, ..., n-1$ for $\mathcal{F}_\sigma$ in the Mode Sampling stage (see Sec. 5.3) to cover all the modes of $\varsigma_1, ..., \varsigma_M$. The probability of failing to sample each ball at least once is no greater than

$$\frac{L}{\varepsilon}\left(1 - \frac{\pi^3\varepsilon^6}{6|C_N|}\right)^{N_P}$$

□

Theorem 2. *Let $\gamma_P : [0, L] \longrightarrow \mathscr{E}$ be a path that connects p_{start} and p_{goal}. Given R as defined by Lemma 2, h_m as defined by equation 5.7, and $\rho \geq 2h_m$, the probability that N_P uniform samples of C_N will fail to yield a path that can lead to a solution is no greater than*

$$\frac{L}{\varepsilon}\left(1 - \frac{\pi^3\varepsilon^6}{6|C_N|}\right)^{N_P} \tag{5.9}$$

where $\varepsilon = \min(R/2, \rho/4)$, and $|C_N|$ is the volume of the necessary condition manifold.

Proof. Using Lemmas 2 and 3, we have established that if we have the conditions $\varepsilon \leq R/2$ and $\varepsilon \leq \rho/4$ where $\rho \geq 2h_m$, it is sufficient to have at least one sample in each ball $B_\varepsilon(p_i), i = 0, ..., n-1$ in order to produce a graph that achieves two properties:

1. The graph contains at least one path from p_{start} to p_{goal} which passes entirely through C_N,

2. The region covered by circles of radius ρ, centered at each vertex along one of the paths from p_{start} to p_{goal} will cover the entirety of $\{\varsigma_1, ..., \varsigma_M\}$.

Therefore, we choose $\varepsilon = \min(R/2, \rho/4)$, and then the probability that one of the balls $B_\varepsilon(p_i)$ will fail to be sampled is no greater than the expression given by equation 5.9. □

Overall Completeness

The success of the Mode Sampling stage requires the Possibility Exploration stage to succeed in finding a viable candidate path. Similarly, the success of the Multi-modal PRM stage requires the Mode Sampling stage to succeed in finding a set of modes that can reach from the start to the goal. Here we prove that the combination of these dependent processes is probabilistically complete given that the individual processes are each probabilistically complete.

Lemma 4. *Consider the randomized processes A and B. Suppose B depends on A such that B can only succeed after A has succeeded. Given $\Pr[\bar{A}] \leq a_F$ and $\Pr[\bar{B}|A] \leq b_F$, then the probability of both processes failing, $\Pr[\bar{A} \cup \bar{B}]$, is no greater than $a_F + b_F$.*

Proof. Define the probability of process A succeeding as $\Pr[A]$ and the probability of it failing as $\Pr[\bar{A}]$. If process B cannot succeed unless process A succeeds, then we know $\Pr[\bar{B}|\bar{A}] \equiv 1.0$ and $\Pr[A|B] \equiv 1.0$. If we are also given $\Pr[\bar{A}] \leq a_F$ and $\Pr[\bar{B}|A] \leq b_F$, we can derive the following:

$$\begin{aligned} \Pr[\bar{A} \cup \bar{B}] &= \Pr[\bar{A}] + \big(1 - \Pr[\bar{A}]\big) \Pr[\bar{B}|A] \\ &\leq a_F + b_F \end{aligned} \tag{5.10}$$

□

Theorem 3. *The probability of the overall process of the w-RPG failing to find a solution will asymptotically converge to zero as the number of samples used for each stage in the process goes to infinity.*

Proof. Consider the Possibility Exploration stage to be process A and the Mode Sampling stage to be process B. From Theorems 1 and 2, we get the following expressions:

$$\begin{aligned} a_F &\leq \frac{L}{\varepsilon}\left(1 - \frac{\pi^3\varepsilon^6}{6|C_N|}\right)^{N_P} \\ b_F &\leq M\left(1 - \frac{r_m^2\Delta\theta_m}{2|\mathcal{F}_\sigma|}\right)^{N_\sigma} \end{aligned} \tag{5.11}$$

We can use the inequality $(1-x) \leq e^{-x}$, for $x \geq 0$ to change these expressions to:

$$\begin{aligned} a_F &\leq \frac{L}{\varepsilon}\exp\left(-\frac{\pi^3\varepsilon^6}{6|C_N|}N_P\right) \\ b_F &\leq M\exp\left(-\frac{r_m^2\Delta\theta_m}{2|\mathcal{F}_\sigma|}N_\sigma\right) \end{aligned} \tag{5.12}$$

Observing that $\alpha_1\exp(-\beta_1) + \alpha_2\exp(-\beta_2) \leq \alpha\exp(-\beta)$ where $\alpha = \alpha_1 + \alpha_2$ and $\beta = \min(\beta_1, \beta_2)$ and combining Lemma 4 with the expressions in equation 5.12, we can get

$$\begin{aligned} \Pr[\bar{A} \cup \bar{B}] &\leq \left(\frac{L}{\varepsilon} + M\right)\exp(-\beta) \\ \beta &= \min\left(\frac{\pi^3\varepsilon^6}{6|C_N|}N_P, \frac{r_m^2\Delta\theta_m}{2|\mathcal{F}_\sigma|}N_\sigma\right) \end{aligned} \tag{5.13}$$

Therefore, as both N_P and N_σ go to infinity, the probability of their combined process failing asymptotically approaches zero, making the combined process probabilistically complete.

This argument can be repeated recursively by viewing the combined process of Possibility Exploration and Mode Sampling as a single process upon which the Multi-modal PRM stage depends. Since Multi-modal PRM is known to be probabilistically complete, adding it as a dependent process onto another probabilistically complete process allows the overall process to still be probabilistically complete. □

Analysis

The expressions which have been derived to prove the probabilistic completeness of the w-RPG also reveal that the multi-stage procedure can offer a better rate of convergence for success than a single-stage procedure would. The parameter ρ is used to restrict the region from which foot placements are sampled during the Mode Sampling stage. This focuses the mode sampling around the candidate route found in the Possibility Exploration stage, ensuring that the samples are conducive toward finding a solution as illustrated in Fig. 5.1. As ρ approaches infinity, the behavior is analogous to eliminating the Possibility Exploration stage altogether and instead merely sampling foot placements uniformly throughout the environment. In this section, we show how the earlier proofs predict an improvement in convergence. We also show simulation results which empirically reinforce this prediction.

Theoretical Analysis

Recall that $\mathcal{F}_\sigma$ is the union of circles with radius ρ, centered around the $\lceil L/\varepsilon \rceil - 1$ vertices of the projected route from the Possibility Exploration stage. This gives us an upper bound on the area covered by $\mathcal{F}_\sigma$:

$$|\mathcal{F}_\sigma| \leq \frac{L}{\varepsilon}\pi\rho^2 \leq 4L\pi\rho$$

Substituting this into equation 5.3 for $\mathcal{F}_\sigma$, we get an upper bound on the likelihood of failure for Mode Sampling in terms of ρ:

$$\Pr[\text{Mode Sampling Failure}] \leq M\left(1 - \frac{r_m^2 \Delta\theta_m}{8L\pi\rho}\right)^{N_\sigma} \tag{5.14}$$

which implies that minimizing ρ will maximize the rate of convergence for the Mode Sampling stage.

However, there are limits to how small ρ can be shrunk for the formula to hold. In particular, the proof for Theorem 1 depends on the assumption that $\mathcal{F}_\sigma$ covers the cylin-

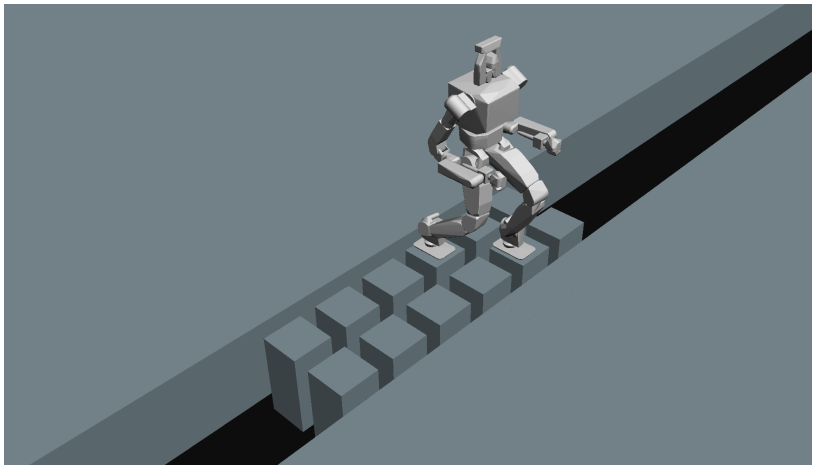
ders associated with the parameters r_i and $\Delta\theta_i$. If $\mathcal{F}_\sigma$ is shrunk to no longer cover those cylinders, then the formula will not hold and we can no longer guarantee probabilistic completeness or asymptotic convergence. To ensure the formulae hold, the proof for Theorem 2 suggests $\rho \geq 2\mathcal{R}_{\text{max}}$ as a lower bound.

It is worth noticing that the formula also predicts the existence of pathological cases which cannot be reliably solved by the w-RPG. Specifically, if r_m or $\Delta\theta_m$ have a value close to zero, then it implies that the solution requires a sample from a manifold with nearly zero volume in SE(2). The probability of randomly sampling a point on such a manifold is close to zero, so we could not expect this approach to reliably work, much like the well-known "narrow passage problem" [64]. There would need to be some additional information provided to the planner that would allow it to find samples on that smaller manifold. For example, [37] used evaluations of the terrain data to adjust infeasible footstep locations. Similarly, an infinitesimal value for R from Lemma 2 would imply that the robot must pass through an extremely narrow passage between obstacles, and would require a potentially infinite number of samples to find the feasible path.

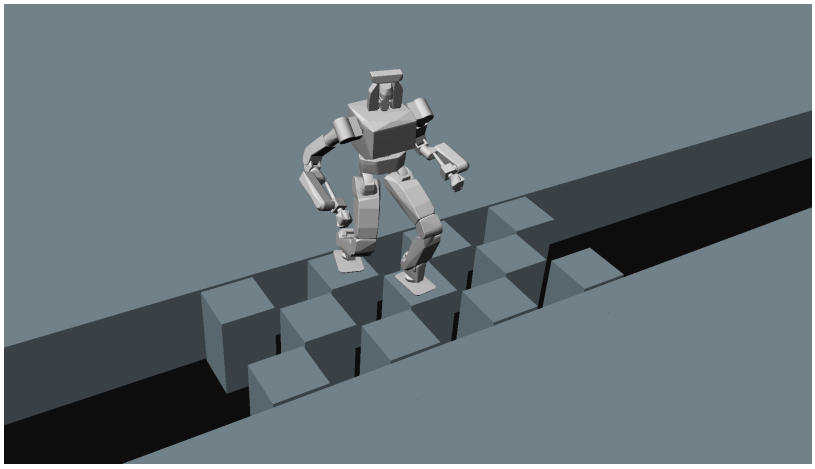
Empirical Results

To empirically test the effects of ρ, we constructed three simple scenarios and ran simulated tests while varying the value of ρ. Screenshots of the scenarios can be seen in Fig. 5.7. A plot of the results is shown in Fig. 5.8. For the "Stepping Stones" and "Checkers" scenarios, the robot needs to find a sequence of foot placements that can get it across a wide gap. In such cases, Mode Sampling is the primary bottleneck, and equation 5.14 plays the dominant role. This gives us performance results which reflect the theoretical predictions of the lower bound.

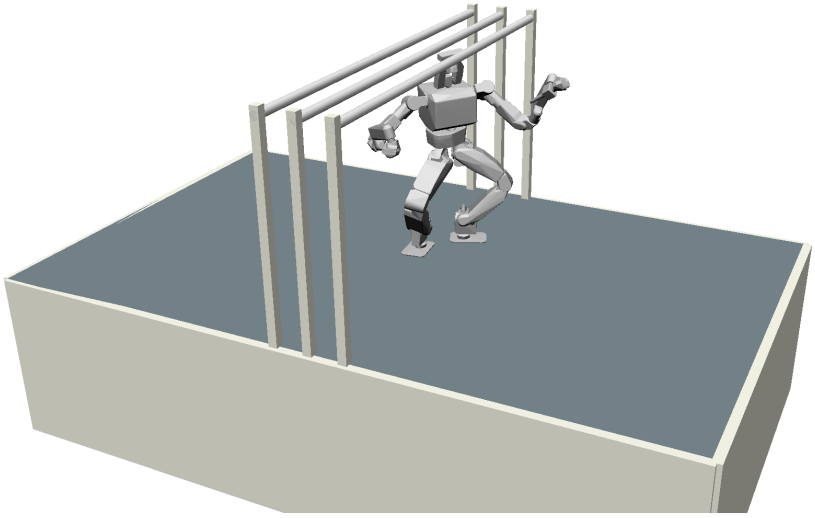
In contrast, the "Pass Under" scenario requires the robot to pass underneath a sequence of three bars. The floor is clear of holes or obstructions, leaving it wide open for the robot to place its feet anywhere. Furthermore, the overall floor space of the environment is relatively



(a) Stepping Stones Scenario



(b) Checkers Scenario



(c) Pass Under Scenario

Figure 5.7: Three scenarios used for simulation tests. In (a) and (b), the robot must get across a gap by taking advantage of narrow stepping stones. In (c), the robot must pass underneath a sequence of bars.

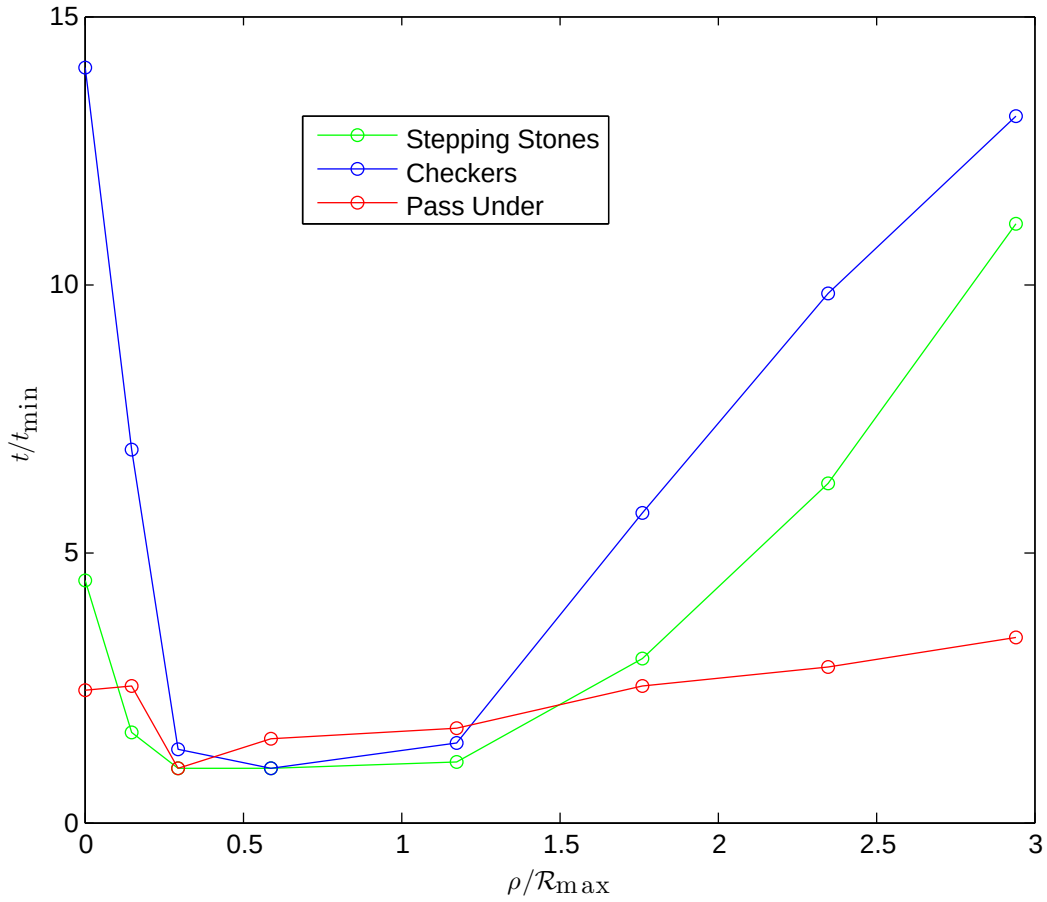


Figure 5.8: Average performance results from three scenarios, illustrating the relationship between ρ and the rate of convergence. The y-axis shows the average time for each scenario, scaled by the data point with the smallest value. The x-axis shows how ρ was varied, scaled by $\mathcal{R}_{\max}$, the furthest distance that the robot is able to step.

small. These factors result in a scenario where Mode Sampling is a less demanding stage. Instead, the physical obstacle of the overhanging bars results in a narrow passage, making R the deciding variable for ε in equation 5.9. Smaller values for ρ may still offer some marginal performance improvements, but it does not appear to be exponential as it is for the other two scenarios.

While the theoretical analysis proposes a value of $\rho = 2\mathcal{R}_{\text{max}}$ to optimize performance while guaranteeing probabilistic completeness, the empirical data suggests that a value in the range $\frac{1}{2}\mathcal{R}_{\text{max}} \leq \rho \leq \mathcal{R}_{\text{max}}$ might be best for performance in practice. A potential strategy could be to schedule the value of ρ so that it begins with a high-performance value and then grows up to the theoretical lower bound over time. It is also plausible that a different proof for Lemma 2 may be able to use a less strict condition than $\rho \geq 2\mathcal{R}_{\text{max}}$.

CHAPTER 6
CONCLUSIONS AND FUTURE WORK

There are considerable challenges inherent to deploying legged robotic platforms on complex terrain. Generating motion plans for legged robots involves finding configuration paths that traverse sequences of kinematic and geometric constraint manifolds with varying dimensionality and narrow intersections. Furthermore, each manifold is determined by a selection of contact points in the environment, and in general there is an infinite continuum of feasible contacts to choose from. In large scale environments with arbitrary geometry, the problem of choosing contact points and finding feasible paths through the resulting constraint manifolds may be especially daunting.

To address these challenges, this thesis presented a combined high- and low-level motion planning structure called the Randomized Possibility Graph (RPG). Along with sets of necessary and sufficient conditions for bipedal walking and a method for quickly sampling feasible foot placements within a limited region, the RPG is able to find feasible bipedal walking trajectories through arbitrary environments without the need for any special geometric representations to facilitate foot placement sampling.

Summary of Contributions

Randomized Possibility Graph: We introduce a method for tackling large-scale motion planning problems in challenging environments. The problem is first addressed at a high-level using the fewest number of dimensions possible. With this high-level representation, we search for a *guide route* rather than computing a whole body motion plan. The edges of the guide route may meet either a set of necessary conditions or a set of sufficient conditions. The satisfaction of these conditions gives us additional insight to the feasibility of the guide route. They also provide a natural way to decompose the overall problem into a

set of parallelizable motion planning queries. These queries are passed to a set of low-level whole body motion planners which can then determine a feasible joint space trajectory that follows along the guide route. While the low-level planners are searching for feasible trajectories, the high-level planner can continue to search for alternative guide routes, in case an easier route can be found. The RPG is a formalization and extension of prior route-based motion planning methods.

Foot Placement Sampling: We present a novel method for the rapid randomized sampling of foot placements in arbitrary 3D environments. The method is shown to work on a variety of geometric features, including disconnected surfaces and curved surfaces. It is fast enough that the time it requires is insignificant in comparison to the overall planning effort. It also alleviates the need for specialized representations of walkable surfaces, such as height maps or well-conditioned meshes.

Bipedal Locomotion Planning: We implemented sets of sufficient and necessary conditions for performing quasi-static bipedal locomotion planning in arbitrary environments. This serves as a proof-of-concept for the RPG to be applied to bipedal locomotion planning. The implementation works on environment models that may use arbitrary geometric representations.

Probabilistic Completeness Analysis: We derive a proof of probabilistic completeness for the RPG when applied to "semi-unstructured" environments. This leads to an analysis which indicates that the RPG can provide an exponential improvement in the overall planner's rate of convergence towards finding a feasible solution. We also provide an analysis of the conditions that are necessary to ensure the improvement in convergence.

Future Work

There are many challenges still remaining before legged robots can be fully effective in real-world deployment.

Dynamics: In completely unstructured environments, there may be obstacles that can only be traversed using dynamic motions, such as jumping or jogging. It should be possible to extend the RPG concept to generate plans for classes of dynamic motions if we can determine sets of necessary and sufficient conditions for those motions. Complex dynamic motions could be evaluated by extending the dimensionality of the high-level stage of the RPG to account for root velocity or system momentum. A kinodynamic motion planner [65] may be used for the high-level planner, $\Gamma_{\mathcal{P}}$, to account for acceleration constraints. For the low-level planners, $\Omega_{\mathcal{M}}$, we may utilize an extreme locomotion planning framework [66] which is designed to convert a root transform and system momentum trajectory (along with a set of available foot placements) into a sequence of feasible dynamic motions. The evaluation of the high-level conditions for these dynamic actions can be interlaced with the evaluation of high-level conditions for quasi-static motions, creating plans that can transition between performing quasi-static and dynamic motions as needed.

Learning: While the current implementation may be applicable to a wide range of environments, its performance is not suitable for real-time deployment except in extremely simple scenarios which could just as easily be solved using a more specialized method. However, the complete autonomy of this method could make it suitable as a basis for a learning algorithm to collect data from. A large set of arbitrary environments could be fed to the RPG to solve, and then a learning algorithm could identify common primitive actions that emerge from the solutions. These learned behaviors could then be collected in a library and recycled by the RPG to improve future performance. This combination of random sampling and learned behaviors could offer a planner that has both high performance

and broad applicability. Since the method does not require a human in the loop, there is minimal overhead in the learning process.

Foot Placement Recognition: The foot placement sampling method presented here is able to identify feasible foot placements that may have eluded prior methods, but there are certain features for which it fails to recognize feasible foot placements. The fundamental shortcoming of the method is that it relies on rejection sampling, which assumes the existence of a continuous region of feasible locations. In reality, there may be corner cases where an exact placement is needed, which cannot be randomly sampled. A more analytical approach to identifying the foot placements may be able to catch such corner cases.

N-Limbs: We expect that the RPG method could extend naturally to robots that can use an arbitrary number of limbs to contact the environment. The principle limitation of applying the RPG to a given problem is the production of necessary and sufficient conditions. It may be possible to automatically learn these conditions by evaluating the system model while performing a variety of primitive actions and identifying inner and outer approximations of the feasibility constraints. This would dramatically reduce the barrier to entry of applying the RPG to arbitrary robot models.

Uncertainty remains a particularly challenging problem in real-world deployment. In this thesis, we assume that full world knowledge is both available and accurate, but these assumptions would be violated during any realistic deployment. It may be possible to design probabilistic conditions that allow the planner to favor choices that have higher certainty associated to them. It may also be possible to use the RPG to generate sets of contingency plans along various alternative routes in case the primary plan turns out to be infeasible.

REFERENCES

[1] M. Hirose and K. Ogawa, "Honda humanoid robots development," *Philosophical Transactions of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*, vol. 365, no. 1850, pp. 11–19, 2007.

[2] Z. Wang and H. Gu, "A review of locomotion mechanisms of urban search and rescue robot," *Industrial Robot: An International Journal*, vol. 34, no. 5, pp. 400–411, 2007. eprint: `http://dx.doi.org/10.1108/01439910710774403`.

[3] G. Pratt and J. Manzo, "The darpa robotics challenge [competitions]," *IEEE Robotics & Automation Magazine*, vol. 20, no. 2, pp. 10–12, 2013.

[4] P. G. De Santos, J. A. Cobano, E Garcia, J Estremera, and M. Armada, "A six-legged robot-based system for humanitarian demining missions," *Mechatronics*, vol. 17, no. 8, pp. 417–430, 2007.

[5] D. Apostolopoulos and J. Bares, "Locomotion configuration of a robust rappelling robot," in *Intelligent Robots and Systems 95.'Human Robot Interaction and Cooperative Robots', Proceedings. 1995 IEEE/RSJ International Conference on*, IEEE, vol. 3, 1995, pp. 280–284.

[6] K. Hauser, T. Bretl, J.-C. Latombe, and B. Wilcox, "Motion planning for a six-legged lunar robot," in *Algorithmic Foundation of Robotics VII*, Springer, 2008, pp. 301–316.

[7] N. A. Radford, P. Strawser, K. Hambuchen, J. S. Mehling, W. K. Verdeyen, A. S. Donnan, J. Holley, J. Sanchez, V. Nguyen, L. Bridgwater, R. Berka, R. Ambrose, M. Myles Markee, N. J. Fraser-Chanpong, C. McQuin, J. D. Yamokoski, S. Hart, R. Guo, A. Parsons, B. Wightman, P. Dinh, B. Ames, C. Blakely, C. Edmondson, B. Sommers, R. Rea, C. Tobler, H. Bibby, B. Howard, L. Niu, A. Lee, M. Conover, L. Truong, R. Reed, D. Chesney, R. Platt, G. Johnson, C.-L. Fok, N. Paine, L. Sentis, E. Cousineau, R. Sinnet, J. Lack, M. Powell, B. Morris, A. Ames, and J. Akinyode, "Valkyrie: NASA's first bipedal humanoid robot," *Journal of Field Robotics*, vol. 32, no. 3, pp. 397–419, 2015.

[8] S. Bartsch, T. Birnschein, F. Cordes, D. Kühn, P. Kampmann, J. Hilljegerdes, S. Planthaber, M. Römmermann, and F. Kirchner, "Spaceclimber: Development of a six-legged climbing robot for space exploration," in *Robotics (ISR), 2010 41st international symposium on and 2010 6th German conference on robotics (ROBOTIK)*, VDE, 2010, pp. 1–8.

[9] G. Nelson, A. Saunders, N. Neville, B. Swilling, J. Bondaryk, D. Billings, C. Lee, R. Playter, and M. Raibert, "Petman: A humanoid robot for testing chemical protective clothing," *ł*, vol. 30, no. 4, pp. 372–377, 2012.

[10] K. Kaneko, K. Harada, F. Kanehiro, G. Miyamori, and K. Akachi, "Humanoid robot hrp-3," in *Intelligent Robots and Systems, 2008. IROS 2008. IEEE/RSJ International Conference on*, IEEE, 2008, pp. 2471–2478.

[11] I.-W. Park, J.-Y. Kim, J. Lee, and J.-H. Oh, "Mechanical design of humanoid robot platform khr-3 (kaist humanoid robot 3: Hubo)," in *Humanoid Robots, 2005 5th IEEE-RAS International Conference on*, IEEE, 2005, pp. 321–326.

[12] S. Rezazadeh, C. Hubicki, M. Jones, A. Peekema, J. Van Why, A. Abate, and J. Hurst, "Spring-mass walking with atrias in 3d: Robust gait control spanning zero to 4.3 kph on a heavily underactuated bipedal robot," in *ASME 2015 dynamic systems and control conference*, American Society of Mechanical Engineers, 2015, V001T04A003–V001T04A003.

[13] S. N. Yadukumar, M. Pasupuleti, and A. D. Ames, "Human-inspired underactuated bipedal robotic walking with amber on flat-ground, up-slope and uneven terrain," in *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, IEEE, 2012, pp. 2478–2483.

[14] M. Raibert, K. Blankespoor, G. Nelson, and R. Playter, "Bigdog, the rough-terrain quadruped robot," *IFAC Proceedings Volumes*, vol. 41, no. 2, pp. 10 822–10 825, 2008.

[15] J. Buchli, J. Pratt, N. Roy, M. P. Murphy, A. Saunders, C. Moreira, A. A. Rizzi, and M. Raibert, "The littledog robot," *The International Journal of Robotics Research*, vol. 30, no. 2, pp. 145–149, 2011.

[16] S. Seok, A. Wang, M. Y. Chuah, D. Otten, J. Lang, and S. Kim, "Design principles for highly efficient quadrupeds and implementation on the mit cheetah robot," in *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, IEEE, 2013, pp. 3307–3312.

[17] P. Hebert, M. Bajracharya, J. Ma, N. Hudson, A. Aydemir, J. Reid, C. Bergh, J. Borders, M. Frost, M. Hagman, J. Leichty, P. Backes, B. Kennedy, P. Karplus, B. Satzinger, K. Byl, K. Shankar, and J. Burdick, "Mobile manipulation and mobility as manipulationdesign and algorithms of robosimian," *Journal of Field Robotics*, vol. 32, no. 2, pp. 255–274, 2015.

[18] J. Lim, I. Lee, I. Shim, H. Jung, H. M. Joe, H. Bae, O. Sim, J. Oh, T. Jung, S. Shin, K. Joo, M. Kim, K. Lee, Y. Bok, D.-G. Choi, B. Cho, S. Kim, J. Heo, I. Kim, J. Lee,

I. S. Kwon, and J.-H. Oh, "Robot system of DRC-HUBO+ and control strategy of team KAIST in DARPA robotics challenge finals," *Journal of Field Robotics*, 2016.

[19] K. Hauser, V. Ng-Thow-Hing, and H. Gonzalez-Baños, "Multi-modal motion planning for a humanoid robot manipulation task," in *Robotics Research*, Springer, 2010, pp. 307–317.

[20] L. P. Kaelbling and T. Lozano-Pérez, "Hierarchical task and motion planning in the now," in *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, IEEE, 2011, pp. 1470–1477.

[21] C. R. Garrett, T. Lozano-Pérez, and L. P. Kaelbling, "Backward-forward search for manipulation planning," in *Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on*, IEEE, 2015, pp. 6366–6373.

[22] M. X. Grey, C. R. Garrett, C. K. Liu, A. Ames, and A. L. Thomaz, "Humanoid manipulation planning using backward-forward search," in *IEEE/RSJ International Conference on Intelligent Robots and Systems, 2016. IROS 2016*, 2016.

[23] J. Pettré, J.-P. Laumond, and T. Siméon, "A 2-stages locomotion planner for digital actors," in *Proceedings of the 2003 ACM SIGGRAPH/Eurographics symposium on Computer animation*, Eurographics Association, 2003, pp. 258–264.

[24] J. Chestnutt, J. Kuffner, K. Nishiwaki, and S. Kagami, "Planning biped navigation strategies in complex environments," in *IEEE Int. Conf. Hum. Rob., Munich, Germany*, 2003.

[25] K. Hauser, T. Bretl, J.-C. Latombe, K. Harada, and B. Wilcox, "Motion planning for legged robots on varied terrain," *The International Journal of Robotics Research*, vol. 27, no. 11-12, pp. 1325–1349, 2008.

[26] M. Zucker, *Learning and Optimization Methods for High Level Planning*. Carnegie Mellon University, 2011.

[27] M. X. Grey, A. D. Ames, and C. K. Liu, "Footstep and motion planning in semi-unstructured environments using randomized possibility graphs," in *ICRA'17. IEEE International Conference on Robotics and Automation, 2017*, IEEE, 2017.

[28] ——, "Probabilistic completeness of randomized possibility graphs applied to bipedal walking in semi-unstructured environments," in *Robotics: Science and Systems*, 2017.

[29] L. E. Kavraki, P. Svestka, J.-C. Latombe, and M. H. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *IEEE transactions on Robotics and Automation*, vol. 12, no. 4, pp. 566–580, 1996.

[30] J. J. Kuffner and S. M. LaValle, "Rrt-connect: An efficient approach to single-query path planning," in *Robotics and Automation, 2000. Proceedings. ICRA'00. IEEE International Conference on*, IEEE, vol. 2, 2000, pp. 995–1001.

[31] N. Perrin, O. Stasse, F. Lamiraux, Y. J. Kim, and D. Manocha, "Real-time footstep planning for humanoid robots among 3d obstacles using a hybrid bounding box," in *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, IEEE, 2012, pp. 977–982.

[32] N. Perrin, O. Stasse, L. Baudouin, F. Lamiraux, and E. Yoshida, "Fast humanoid robot collision-free footstep planning using swept volume approximations," *IEEE Transactions on Robotics*, vol. 28, no. 2, pp. 427–439, 2012.

[33] S. Tonneau, N. Mansard, C. Park, D. Manocha, F. Multon, and J. Pettré, "A reachability-based planner for sequences of acyclic contacts in cluttered environments," in *International Symposium on Robotics Research (ISSR 2015)*, 2015.

[34] Y. Shimizu and T. Sugihara, "Efficient path planning of humanoid robots with automatic conformation of body representation to the complexity of environments," in *Humanoid Robots (Humanoids), 2012 12th IEEE-RAS International Conference on*, IEEE, 2012, pp. 755–760.

[35] J. J. Kuffner, K. Nishiwaki, S. Kagami, M. Inaba, and H. Inoue, "Footstep planning among obstacles for biped robots," in *Intelligent Robots and Systems, 2001. Proceedings. 2001 IEEE/RSJ International Conference on*, IEEE, vol. 1, 2001, pp. 500–505.

[36] J. Kuffner, S. Kagami, K. Nishiwaki, M. Inaba, and H. Inoue, "Online footstep planning for humanoid robots," in *Robotics and Automation, 2003. Proceedings. ICRA'03. IEEE International Conference on*, IEEE, vol. 1, 2003, pp. 932–937.

[37] J. Chestnutt, K. Nishiwaki, J. Kuffner, and S. Kagami, "An adaptive action model for legged navigation planning," in *Humanoid Robots, 2007 7th IEEE-RAS International Conference on*, IEEE, 2007, pp. 196–202.

[38] K. Hauser and J.-C. Latombe, "Multi-modal motion planning in non-expansive spaces," *The International Journal of Robotics Research*, 2009.

[39] K. Hauser and V. Ng-Thow-Hing, "Randomized multi-modal motion planning for a humanoid robot manipulation task," *The International Journal of Robotics Research*, vol. 30, no. 6, pp. 678–698, 2011.

[40] D. Hsu, J.-C. Latombe, and R. Motwani, "Path planning in expansive configuration spaces," in *Robotics and Automation, 1997. Proceedings., 1997 IEEE International Conference on*, IEEE, vol. 3, 1997, pp. 2719–2726.

[41] R. Kindel, D. Hsu, J.-C. Latombe, and S. Rock, "Kinodynamic motion planning amidst moving obstacles," in *Robotics and Automation, 2000. Proceedings. ICRA'00. IEEE International Conference on*, IEEE, vol. 1, 2000, pp. 537–543.

[42] L. Sentis and O. Khatib, "A whole-body control framework for humanoids operating in human environments," in *Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006.*, IEEE, 2006, pp. 2641–2648.

[43] T. Sugihara and Y. Nakamura, "Whole-body cooperative balancing of humanoid robot using cog jacobian," in *Intelligent Robots and Systems, 2002. IEEE/RSJ International Conference on*, IEEE, vol. 3, 2002, pp. 2575–2580.

[44] M. Gienger, H. Janssen, and C. Goerick, "Task-oriented whole body motion for humanoid robots," in *5th IEEE-RAS International Conference on Humanoid Robots, 2005.*, IEEE, 2005, pp. 238–244.

[45] D. Dubois and H. Prade, *Possibility theory: An approach to computerized processing of uncertainty*. Springer Science & Business Media, 2012.

[46] D. Berenson, S. Srinivasa, D. Ferguson, and J. Kuffner, "Manipulation planning on constraint manifolds," in *IEEE International Conference on Robotics and Automation (ICRA '09)*, 2009, pp. 625–632.

[47] T. Kunz and M. Stilman, "Manipulation planning with soft task constraints," in *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, IEEE, 2012, pp. 1937–1942.

[48] R. Deits and R. Tedrake, "Footstep planning on uneven terrain with mixed-integer convex optimization," in *2014 IEEE-RAS International Conference on Humanoid Robots*, IEEE, 2014, pp. 279–286.

[49] T. Bretl and S. Lall, "Testing static equilibrium for legged robots," *IEEE Transactions on Robotics*, vol. 24, no. 4, pp. 794–807, 2008.

[50] J. Pan, S. Chitta, and D. Manocha, "Fcl: A general purpose library for collision and proximity queries," in *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, IEEE, 2012, pp. 3859–3866.

[51] T. Bretl, S. Lall, J.-C. Latombe, and S. Rock, "Multi-step motion planning for free-climbing robots," in *Algorithmic Foundations of Robotics VI*, Springer, 2004, pp. 59–74.

[52] K. Hauser, T. Bretl, and J.-C. Latombe, "Learning-assisted multi-step planning," in *Robotics and Automation, 2005. ICRA 2005. Proceedings of the 2005 IEEE International Conference on*, IEEE, 2005, pp. 4575–4580.

[53] C. L. Nielsen and L. E. Kavraki, "A two level fuzzy prm for manipulation planning," in *Intelligent Robots and Systems, 2000.(IROS 2000). Proceedings. 2000 IEEE/RSJ International Conference on*, IEEE, vol. 3, 2000, pp. 1716–1721.

[54] G. Sánchez and J.-C. Latombe, "On delaying collision checking in prm planning: Application to multi-robot coordination," *The International Journal of Robotics Research*, vol. 21, no. 1, pp. 5–26, 2002.

[55] I. Mordatch, E. Todorov, and Z. Popović, "Discovery of complex behaviors through contact-invariant optimization," *ACM Transactions on Graphics (TOG)*, vol. 31, no. 4, p. 43, 2012.

[56] M. Zucker, N. Ratliff, A. D. Dragan, M. Pivtoraiko, M. Klingensmith, C. M. Dellin, J. A. Bagnell, and S. S. Srinivasa, "Chomp: Covariant hamiltonian optimization for motion planning," *The International Journal of Robotics Research*, vol. 32, no. 9-10, pp. 1164–1193, 2013.

[57] N. Ratliff, M. Zucker, J. A. Bagnell, and S. Srinivasa, "Chomp: Gradient optimization techniques for efficient motion planning," in *ICRA'09. IEEE International Conference on Robotics and Automation, 2009*, IEEE, 2009, pp. 489–494.

[58] D. Berenson and S. S. Srinivasaz, "Probabilistically complete planning with end-effector pose constraints," in *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, IEEE, 2010, pp. 2724–2730.

[59] A. D. Ames, R. Vasudevan, and R. Bajcsy, "Human-data based cost of bipedal robotic walking," in *Proceedings of the 14th international conference on Hybrid systems: Computation and control*, ACM, 2011, pp. 153–162.

[60] J. P. Reher, A. Hereid, S. Kolathaya, C. M. Hubicki, and A. D. Ames, "Algorithmic foundations of realizing multi-contact locomotion on the humanoid robot durus," 2016.

[61] J. J. Kuffner, "Effective sampling and distance metrics for 3d rigid body path planning," in *Robotics and Automation, 2004. Proceedings. ICRA'04. 2004 IEEE International Conference on*, IEEE, vol. 4, 2004, pp. 3993–3998.

[62] L. E. Kavraki, M. N. Kolountzakis, and J.-C. Latombe, "Analysis of probabilistic roadmaps for path planning," *IEEE Transactions on Robotics and Automation*, vol. 14, no. 1, pp. 166–171, 1998.

[63] P. Svestka, *On probabilistic completeness and expected complexity of probabilistic path planning*. Universiteit Utrecht, Faculty of Mathematics & Computer Science, 1996.

[64] Z. Sun, D. Hsu, T. Jiang, H. Kurniawati, and J. H. Reif, "Narrow passage sampling for probabilistic roadmap planning," *IEEE Transactions on Robotics*, vol. 21, no. 6, pp. 1105–1115, 2005.

[65] S. M. LaValle and J. J. Kuffner Jr, "Randomized kinodynamic planning," *The International Journal of Robotics Research*, vol. 20, no. 5, pp. 378–400, 2001.

[66] C. M. Dellin and S. S. Srinivasa, "A framework for extreme locomotion planning," in *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, IEEE, 2012, pp. 989–996.