RICE UNIVERSITY

A Unifying Framework for Constrained Sampling-Based Planning

by

Zachary Kingston

A THESIS SUBMITTED
IN PARTIAL FULFILLMENT OF THE
REQUIREMENTS FOR THE DEGREE

Master of Science

APPROVED, THESIS COMMITTEE:

Lydia Kavraki

Lydia E. Kavraki, Chair
Noah Harding Professor of Computer Science

[illegible]

Anshumali Shrivastava
Assistant Professor of Computer Science

[illegible]

Marcia O'Malley
Professor of Mechanical Engineering

[illegible]

Mark Moll
Senior Research Scientist

Houston, Texas

December, 2017

ABSTRACT

A Unifying Framework for Constrained Sampling-Based Planning

by

Zachary Kingston

Complex robots with many degrees-of-freedom (e.g., humanoids, mobile manipulators) have been increasingly applied to achieve tasks in fields such as disaster relief or spacecraft logistics. Finding motions for these systems autonomously is necessary if they are to be applied in unstructured environments not known *a priori*, as they must compute motions on-the-fly. Sampling-based motion planning algorithms have been shown to be effective for finding motions for high-dimensional systems such as these. However, the problems these robots face typically take the form of tasks with *constraints*, which limit the valid motions a robot can take (e.g., turning a valve about its axis, carrying a tray with both arms, etc.). Incorporating constraints while planning introduces significant challenges, as constraints induce a lower-dimensional manifold of constraint-satisfying configurations within the robot's configuration space. The lower-dimensional structure of the manifold throws a wrench into the basic operation of a sampling-based planner, necessitating a *constraint methodology* to provide a means for the planner to satisfy constraints.

Within the literature, many constrained sampling-based motion planning methods have been proposed for sampling-based planning with constraints. Each of these methods introduces a constraint methodology of their own to tackle the issues raised when considering constraints. This thesis organizes several previously proposed constraint methodologies along of a spectrum, cataloged by the amount of bookkeeping and computation used to approximate the manifold of constraint-satisfying configurations. Notably, previous constrained sampling-based methods augment a single sampling-based algorithm with their constraint methodology to create a bespoke planner.

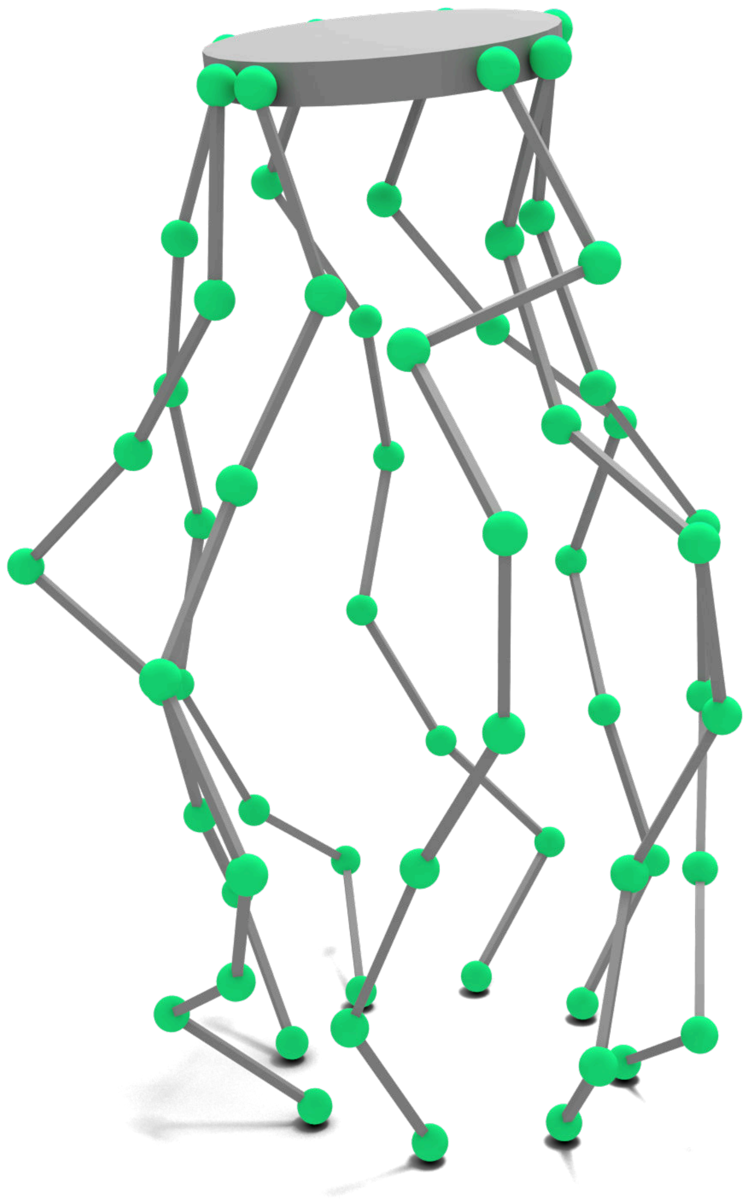
This thesis presents a general framework for sampling-based motion planning with geometric constraints, unifying prior works by approaching the constrained motion planning problem at a higher level of abstraction. The framework decouples the constraint methodology from the planner's method for exploration by presenting the constraint-induced manifold as a configuration space to the planner, hiding details of the constraint methodology behind the space's primitive operations. Three constraint methodologies from the literature are emulated within the framework. The framework is demonstrated with a range of planners using the three emulated constraint methodologies in a set of simulated problems. Results show the advantages decoupling brings to constrained sampling-based planning, with novel combinations of planners and constraint methodologies surpassing emulated prior works. The framework is also easily extended for novel planners and constraint spaces.

Acknowledgments

I find myself hard-pressed to fully communicate how much so many people have meant to me in my graduate pursuit. Lydia and Mark are a constant support, and their incredible advisement and faith in me since my undergraduate years inspires deep feelings of thankfulness. I could not ask for better advisors, and I also thank the entirety of my committee for their time and efforts.

My family, Sally, Alayna, Amy, and Michael have been endlessly supportive through everything I have done, even when I myself am scarce. My housemates Oliver, Logan, and Eric, are staunch friends and a grounding presence even when we are awash with our responsibilities. Susan, I am so fortunate to have such a caring girlfriend. I treasure the time we can make for each other even when we are at our busiest. Thank you to my fellow lab-mates, Keliang, Andrew, Bryce, Cannon, and everyone in the Kavraki Lab for working with me, and being friends. I also would like to thank the NASA cohort, Logan and Julia, for the wonderful opportunity they afforded me with Robonaut 2 and the inspiration it brought me in my work.

Finally, preliminary work on the framework was done by Caleb Voss [1]. My work has been supported in part by NSF IIS 1317849, NSF IIS 1718478, Rice University funds, and by NASA Space Technology Research Fellowship 80NSSC17K0163.



"Holonomic constraints do not fundamentally change the path planning problem. They will only be examined briefly in Section 2."

— J.-C. Latombe, *Robot Motion Planning*, Chapter 9 [2]

Contents

Illustrations

Chapter 1

Introduction

Consider a spacecraft, where resources are limited, creating need for a system can reuse human tools and spaces. Robonaut 2 (R2) is a high-dimensional robotic platform designed to work in the same space as humans in spacecraft (shown in Figure 1.1). Future NASA mission concepts also require spacecraft to be uncrewed for extended periods of time, but systems inside still require maintenance. R2, and other robots, are faced with complex *constrained* tasks such as removing a bag from a rack, turning a valve, and executing closed chain motions with both feet down. In order for a robot to operate autonomously, it requires a *motion planning* system that can generate movements from high-level descriptions.

Motion planning for articulated robots has seen significant advances, and is an essential tool for a robotic system with any level of autonomy. With planning, a robot's movements can be specified with start and goal configurations, rather than a full prescription of intermediate states [2, 3]. However, there is an increasing need for plans that satisfy constraints on a robot's configuration, as constraints can concisely specify complex motions for a robot. In some instances, such as some specific end-effector pose constraints, effective solutions exist for some cases, but in general finding motions for complex robots subject to general configuration constraints poses significant challenges. Such constraints are ubiquitous in robot manipulation (e.g., opening a door, sliding a drawer, pushing an object, etc.).

In general, sampling-based planners have been effective at planning motions for high-dimensional systems [3]. These planners randomly explore the robot's configuration space and build a discrete representation of valid motions. Many sampling-based planners have

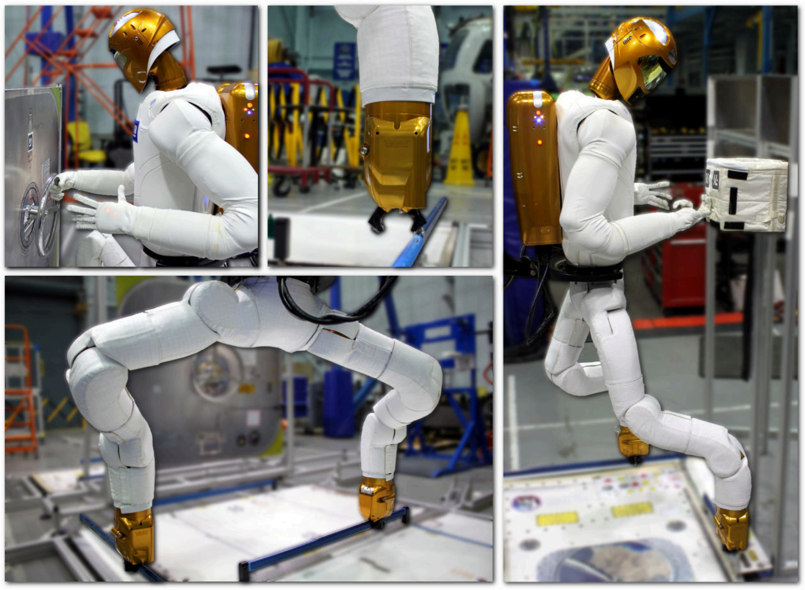


Figure 1.1 : Examples of NASA's Robonaut 2 (R2), a dexterous humanoid robot, executing constrained tasks. Clockwise from top-left, R2 turns a valve, docks with a handrail, extracts a bag from a rack, and moves its torso with both feet attached to handrails. Credit: NASA.

been developed with different methods to explore and exploit the valid motions of a robot. However, incorporating constraints in planning is still difficult, as finding configurations that satisfy constraints is challenging. Recently, several algorithms have been developed for planning with constraints that are effective for realistic problems [4, 5, 6]. However, these algorithms are somewhat limited in the sense that they adapt a specific sampling-based algorithm to also satisfy task constraints, convolving constraint satisfaction with planning methodology.

1.1 Contribution

To understand the contribution of this thesis, it is important to understand the structure of constrained sampling-based planners. Constrained sampling-based planning algorithms presented in the literature differ primarily by their *constraint methodology*, or technique applied to generate feasible motion and samples that satisfies constraints. These methodologies can be placed upon a spectrum, delineated by the amount of computation and bookkeeping they perform to satisfy constraints. Each of these methodologies necessarily makes trade-offs between space and time efficiency, performing well in some environments but potentially failing in others. None of the constrained sampling-based planners in the literature make dramatic changes to the structure of the underlying augmented motion planning algorithm, these methods instead augment primitive operations to generate feasible motion. Additionally, just as trade-offs are made in constrained planners with constraint methodologies, there are many unconstrained sampling-based planners each with their own heuristics or exploration strategies to perform well in certain environments. However, as the state-of-the-art stands now, developing a constrained sampling-based planner with a choice of constraint methodology well-suited to the constraint and an exploration strategy well-suited to the problem requires design of a bespoke planner that integrates the two.

This thesis presents a solution to the design of constrained sampling-based algorithms for more complex systems by means of a framework that decouples constraint satisfaction from space exploration in the planner. The framework is a layer over the unconstrained planning problem, integrating the constraint methodology at the level of the *space* the planner plans within, rather than the planner itself. With this framework, a broad class of sampling-based planners can utilize many previously proposed constraint satisfaction methods and leverage the tools developed by the community, such as asymptotically optimal planners [7], path optimization [8], or domain specific planners for high-dimensional problems [9]. The conceptual framework encapsulates and extends previous approaches in the literature by approaching the problem from a higher-level of abstraction. Presented in this thesis are emulations of three successful and widely known methodologies: CBIRRT2 [4], TB-RRT [6], and AtlasRRT [5]. Additionally, different constraint satisfaction methodologies can all use the same underlying constraint representation within the framework. Furthermore, experimental results show that different problems can be solved more successfully using novel combinations of planning algorithms and implicitly defined constrained spaces.

The framework was originally presented in the Internation Symposium of Robotics Research 2017 [10], and is a first step towards broad application of sampling-based planning techniques for high-dimensional systems, as it enables high-dimensional and domain-specific planning algorithms to handle tasks with constraints. Robonaut 2, and other systems like it, will no longer need custom planners [11] with the framework, and instead can leverage advanced sampling-based planning techniques that exist in the literature.

1.2 Organization

The thesis is organized as follows. Chapter 2 formulates and discusses the mathematical objects and constructs necessary to define the constrained motion planning problem. Chapter 3 presents and discusses related work, and organizes prior approaches to constrained sampling-based planning along a spectrum. The framework is presented at a high-level in Chapter 4, along with the details of the three constraint methodologies emulated within the framework. Empirical results and implementation details of the framework are shown in Chapter 5. Finally, further discussion of the framework and concluding remarks are given in Chapter 6.

Chapter 2

Problem Formulation

This chapter provides the notation and mathematical constructs necessary for constrained sampling-based motion planning. Motion planning, particularly constrained motion planning, draws from concepts in differential geometry to describe the various spaces utilized in the planning process. A good reference for these topics is [12].

This chapter is organized as follows. First, Section 2.1 describes the unconstrained instance of the motion planning problem. Second, Section 2.2 describes the constrained motion planning problem. Finally, some techniques used for satisfying constraints are provided in Section 2.3.

2.1 The Motion Planning Problem

A key idea in motion planning is to transform the problem of planning for a dimensioned, articulate robot into planning for a single point that represents the robot in a higher-dimensional space. This space is called the configuration space.

Definition 2.1. CONFIGURATION SPACE

A *configuration* of the robot is denoted by $q \in \mathcal{Q}$, where $\mathcal{Q}$ is the *configuration space*, a metric space and differentiable manifold. The configuration space contains all configurations of the robot, whether they are valid and collision-free or not. The free portion of the configuration space $\mathcal{Q}_{free} \subseteq \mathcal{Q}$ is the *free space*: the set of all configuration where the robot is not ion collision with obstacles or itself.

For example, an abstract point robot on the plane is parameterized by the x- and y-value that describes its location and many manipulators are parameterized by the position of the revolute joints of the arm. The number of degrees of freedom of a robot, i.e., the dimensionality of its configuration space, is denoted by n. Given a configuration space, the motion planning problem is defined as follows.

Definition 2.2. MOTION PLANNING PROBLEM
The *motion planning problem* is defined as finding a continuous, collision-free path from q_{start} to q_{goal} in configuration space $\tau : [0,1] \rightarrow \mathcal{Q}_{free}$, $\tau(0) = q_{start}$, $\tau(1) = q_{goal}$.

In many cases, avoiding collisions is the only concern for computing a valid path. One of the most powerful features of sampling-based planners is that they avoid explicit computation of $\mathcal{Q}_{free}$, as described in Chapter 3.4. Additional details on the formulation of the motion planning problem and the configuration space can be found in [2].

2.2 The Constrained Motion Planning Problem

To discuss constrained motion planning, the types of constraints that are considered must first be discussed. Most commonly, specialized constraints take the form of end-effector constraints. As the end-effector is the component of the robot carrying out the task, it is desired to be constrained according to some objective, whether that be maintaining contact, not rotating past a certain limit, or otherwise. Recently, many approaches have been taken in the literature to specify the motion constraints in a general formulation, presented below.

Definition 2.3. CONSTRAINT FUNCTION
A *constraint function* $F(q) : \mathcal{Q} \rightarrow \mathbb{R}^k$ evaluates to $F(q) = \mathbf{0}$ when q satisfies the constraint. This work assumes F is continuous (C^1) and differentiable everywhere. k is the number of equality constraints imposed.

For this work, the constraints are purely geometric, and rely only on the configuration of the robot q, not on other properties of the robot's motion such as velocities or accelerations. Constraints of this type are *holonomic*, or integrable constraints. This class of constraints captures a broad and interesting class of possible task encodings, covered in detail in Chapter 3.2. For a brief example, end-effector constraints and balance constraints can both be formulated as a constraint function.

The constraint function defines a lower-dimensional implicit constrained configuration space within the ambient configuration space.

Definition 2.4. CONSTRAINT MANIFOLD
The constraint function defines an $(n-k)$-dimensional implicit constrained configuration space within the ambient configuration space, the *constraint manifold*:

$$\mathcal{X} = \{\, q \in \mathcal{Q} \mid F(q) = \mathbf{0} \,\}$$

As F is continuous and differentiable everywhere, $\mathcal{X}$ is a differentiable manifold. k, the number of equality constraints, is referred to as the *co-dimension* of the constraint manifold with respect to the configuration space. $\mathcal{X}$ is also referred to as the implicit manifold, or the manifold for short.

The relative measure (volume) of $\mathcal{X}$ compared to $\mathcal{Q}$ is small and usually 0, given its lower-dimensionality. Similar to the definition of $\mathcal{Q}_{free}$, define $\mathcal{X}_{free} \subseteq \mathcal{X}$ as all collision-free configurations that also satisfy constraints, $\mathcal{X}_{free} = \mathcal{X} \cap \mathcal{Q}_{free}$.

Definition 2.5. CONSTRAINED MOTION PLANNING PROBLEM
The *constrained motion planning problem*, with a constraint function F and configuration space $\mathcal{Q}$, is a problem of finding a continuous, collision-free path $\tau : [0,1] \to \mathcal{X}_{free}$.

For example, consider a point robot with a configuration space $\mathcal{Q} \subset \mathbb{R}^3$. Given $F(q) =$

$||q|| - 1$, the robot is constrained to the surface of a unit sphere, a two-dimensional manifold in $\mathbb{R}^3$.

Note that inequality constraints of the form $F(q) \leq \mathbf{0}$ can be dealt with in much the same way as collision-avoidance is dealt with, as collision-avoidance is can be phrased as an inequality constraint (e.g., halfplanes to define polytope obstacle).

2.3 Satisfying Constraints

A critical element of a constrained planning algorithm is its constraint methodology, or how it generates configurations that satisfy the constraint. This section covers some of the tools employed by constrained sampling-based planning algorithms described in Chapter 3.5.

Definition 2.6. PROJECTION OPERATOR
A *projection operator* is a continuous idempotent mapping $P(q) : \mathcal{Q} \to \mathcal{X}$, where if $q \in \mathcal{X}, P(q) = q$.

Projection takes a configuration and *projects* it onto the surface of the implicit manifold, solving for a root of the constraint function. Typically this is implemented using Jacobian gradient descent, using the Jacobian of the constraint function $J(q)$. The next step in the descent Δq can be computed by solving the system of equations:

$$J(q)\Delta q = F(q),$$

either through pseudo-inverse techniques or other methods. The descent stops when $F(q) = \mathbf{0}$. A more comprehensive look at projection for constrained motion planning is found in [4]. More contextual details of projection for constrained motion planning is given in Chapter 3.5.2.2.

Local parameterization of the implicit manifold can be accomplished by a *tangent space* (alternatively, a *chart* from [5]). The tangent space is constructed by finding the basis for the nullspace of $J(q)$, which can be computed through a matrix decomposition.

Definition 2.7. TANGENT SPACE
The *tangent space* T_q is a $(n-k)$-dimensional space with its origin at a configuration $q \in \mathcal{X}$, with an $n \times (n-k)$ orthonormal basis Φ_q.

A point $t \in T_q$ can be mapped into $q_t \in \mathcal{Q}$ by $q_t = q + \Phi_q t$. To map the configuration q_t onto the manifold (an exponential map), an orthonormal projection can be computed by solving the system of equations:

$$F(q) = \mathbf{0} \qquad \text{and} \qquad \Phi_q^T(q - q_t) = \mathbf{0}$$

The opposite mapping from the manifold to T_q is much simpler: $t = \Phi_q^T(q - q_t)$. Tangent spaces are composed into an approximation of the manifold by the AtlasRRT and TB-RRT constrained spaces within the framework. A more comprehensive look at manifold approximation for planning can be found in [5], with many operations on implicit manifold discussed in [13]. The concepts of differential manifolds outlined here are covered in depth in [12]. More contextual details of tangent spaces and atlas construction for constrained motion planning are given in Chapters 3.5.2.3 and 3.5.2.4.

Chapter 3

Related Work

This thesis is focused on studying sampling-based planning with geometric task constraints [14], which has a wide breadth of literature concerning both techniques to plan motions and represent constraints. While sampling-based planning is the dominant motion planning paradigm today for highly articulated robots, several different classes of motion planning algorithms have been developed over the years. This chapter organizes and discusses a breadth of literature, first introducing early planning methods in Section 3.1. Next, a brief overview of constraints for constrained motion planning is presented in Section 3.2. Constrained motion planning algorithms that are non-sampling methods are presented in Section 3.3. Then, unconstrained sampling-based algorithms are presented in Section 3.4. This leads into the final section which presents and organizes constrained sampling-based planners on a spectrum, Section 3.5.

3.1 Early Methods

Motion planning is a PSPACE-hard problem [15], with complexity growing with the number of the robot's degrees-of-freedom. Although exact algorithms exist, they are difficult to implement and scale poorly to high-dimensional robots. Early on, potential fields were proposed as an alternative to exact methods where following the gradient of the potential would guide a robot to its goal [16]. It is difficult, though, to come up with a general mechanism to escape local minima of a potential function [17] or design one that has only one minimum. Heuristic search techniques that operate over a discretization of the space of all robot configurations are yet another family of planning algorithms. These algorithms provide so-called *resolution completeness*: a path will be found if the discretization is fine enough [14]. A careful choice of resolution and heuristics are critical for heuristic search. In principle, many of the classes of motion planning algorithms described above could be adapted to incorporate constraints (e.g., discretized search with constraints [18]). However, due to the complications of scaling these methods to higher-dimensional systems, they are generally not applied to modern systems.

Sampling-based algorithms (the focus of this thesis) take a very different approach. They randomly sample valid configurations and form a graph of valid motions [3]. Sampling-based planning algorithms typically provide *probabilistic completeness*: the probability of finding a solution goes to 1 with the run time of the algorithm, provided a solution exists [14]. Sampling-based motion planning algorithms have been shown to be effective at solving motion planning problems in a broad range of settings with minimal changes. They have also been used in very different contexts, such as computer graphics and computational structural biology (see, e.g., [19, 20]).

Constraints on motion have had a rich history in industrial control, specifying Cartesian end-effector constraints to describe assembly tasks [21, 22, 23]. For modern robotics, constraints can be used to specify many useful manipulation tasks, intrinsic properties of the robot such as parallel manipulators and closed chains and even problems outside of robotics, such as structural biology [24]. The first applications of geometric constraints to planning in low-dimensional spaces were reduced to problems of finding geodesics on polyhedral structures [25], similar to finding shortest paths of visibility graphs [26, 27]. Additionally, as motion planning was applied to more complex, higher-dimensional robotic systems, geometric constraints increased the difficulty of the motion planning problem and required additional consideration to plan effectively.

3.2 Constraint Expression

As stated in Chapter 2.2, end-effector constraints are a common encoding of task constraints. End-effector constraints, while not all-encompassing, have the benefit of being intuitive to specify by a user of a robotic system, especially for generating novel constraints while attempting to execute some task. End-effector constraints have origins in industrial control with Cartesian constraints between interacting objects [21], which were developed into end-effector constraints on manipulators [22, 23].

There are many modern incarnations of end-effector constraints [28], such as the *Task-Space Region* formulation employed by the CBIRRT2 planner [4]. Task-space regions are general method for sampling configurations subject to many Cartesian end-effector constraints, but not all (such as a screwing motion). End-effector constraints can also be extended to closed-chain systems, by decomposing the closed chain into two manipulators that must maintain contact of the end-effectors throughout the entire motion, closing the chain [29]. Task-space regions have also been extended to *Task-Space Region Chains* [4], which model articulated kinematic structures such as doors and drawers in a scene for end-effector constraints, similar to how closed-chain systems are planned for.

Recently, many approaches have been taken in the literature to specify the motion constraints of the form $F(q) = \mathbf{0}$ on a robotic system. The most general approach is to not assume any properties of the constraint function in relation to the kinematic structure of the robot [5]. This general representation comes at the price of the ability to exploit features of the constraint that might be employed by a system utilizing end-effector constraints. If constraints can be cast as functions that can be automatically differentiated or have analytic derivatives, the solver speed can be improved and satisfying configurations can be generated faster [30]. This thesis focuses on constraints of this form, as this formulation is general and encompasses most constraints that are of concern.

3.2.1 Constraint Composition

The representation of a constraint function and its derivative is critical to efficiently solving an instance of a constraint motion planning problem. This becomes challenging when a constraint function is a composition of many constraints. So far the discussion of constraints has limited itself to single constraint functions, which encode the full set of constraints on a system. A question of interest for more complex systems is how to compose multiple constraints on a system into one coherent function.

Composing constraints that all need to be simultaneously satisfied into one constraint function is non-trivial. For example, given a humanoid robot, what is the best way to combine and encode balance, the task objective, a visibility region that must be maintained, and others? This “and”-ing of constraints together into one function can be thought of as computing the intersection of sets of configurations that satisfy each constraint. When the structure of the constraints is known and their importance and dependence can be deduced, it is possible to order the constraints and use nullspace projection to attempt to solve hierarchically [31], as is done in [11]. Another approach is to use more advanced gradient descent techniques or cyclic projection such as those discussed in [4]. The method of constraint composition when considering multiple simultaneous constraints has dramatic effects on performance, and method selection is critical to efficient operation of the planner and the completeness of the method.

What if the composition of tasks has more than one modality? Intermittent contact, an essential component of manipulation and legged locomotion, requires the constant addition and removal of constraint [32, 33]. Combining constraints where only a subset need to be satisfied at any given time is an “or”-ing of the constraints together: creating a union of constraint manifolds that potentially intersect and overlap. This creates singularity points, changes in dimension, and other problematic changes that most constraint methodologies cannot handle. One approach is to use a higher-level discrete representation of a “graph” to handle mode switching between different constraints, which might have different numbers of equality constraints imposed on the system [30, 34]. This problem can also be thought of as a special instance of hierarchical planning, with a discrete selection of constraint modality followed by geometric constrained planning. Foot-step planning and other task and motion planning problems can all be thought of within this framework [35, 36, 37, 38, 39]. Each of these planners employs domain specific knowledge to solve the problem efficiently, but no general purpose solutions have been proposed.

3.3 Non-Sampling Methods

In recent years, many non-sampling methods have also been proposed that can scale to complex systems. Although not the focus of this thesis, a summary is provided for completeness.

As robotic manipulators became more complex and had degrees-of-freedom redundant to the task at hand, Cartesian curve tracking required resolution of the redundant degrees-of-freedom of the robot [40], solved using inverse kinematic (IK) techniques [41, 42]. Finding configurations that satisfy hard constraints can be handled in a variety of ways. In common cases, such as an end effector pose constraints, IK solvers can be used. IK is concerned with the problem of finding robot configurations such that an end effector achieves a desired pose. For more complex constraints or in the presence of kinematic redundancies, IK alone is typically not sufficient to guarantee completeness. One approach to the constrained planning problem is to plan in the robot’s workspace, so geometric constraints can be directly evaluated and satisfying poses can be sampled. Post-planning a path in the robot’s configuration space is generated using IK similar to curve tracking technqiues [31, 43]. However, these methods may not be efficient as re-planning is required if a found path cannot be mapped into the configuration space of the robot. Completeness is also not guaranteed unless all feasible IK solutions can be generated given the constraints.

Another approach that operates within the robot's workspace is reactive control, which uses convex optimization to find local satisfying motions, such as those used at the DARPA Robotics Challenge (e.g., [44, 45, 46]). While effective with operator supervision, these controllers are usually incomplete and risk local minima. As local controllers are optimization-based methods, hard constraints are relaxed into soft constraints, and invalid motions can be generated.

One possible alternative to sampling-based planning is to use penalty functions to turn hard constraints into soft constraints and use trajectory optimization methods [47, 48, 49, 50] instead. Trajectory optimization approaches optimize within trajectory space and are effective for everyday manipulation tasks, but suffer from many of the same shortfalls as reactive control. When combining optimization with random trajectory initialization in the appropriate functional space, probabilistic completeness can still be preserved [47]. Comprehensive comparison of constrained non-sampling-based methods to sampling-based planners has not been done, and a thorough analysis is left as future work.

3.4 Unconstrained Sampling Methods

It is helpful to first describe the general structure of (unconstrained) sampling-based planning algorithms and the common primitives they rely on. For a more in-depth review of sampling-based planning see [3, 14, 51]. The general idea behind sampling-based planning is to avoid computing the free space exactly, but, instead, sample free configurations and connect them to construct a tree/graph that approximates the connectivity of the underlying free

```
procedure GraphPlanner($q_{\text{start}}$, $q_{\text{goal}}$)
    $\mathcal{G}$.init($q_{\text{start}}$, $q_{\text{goal}}$);
    while no path from $q_{\text{start}}$ to $q_{\text{goal}}$ do
        $q_{\text{rand}} \leftarrow$ Sample();
        $Q \leftarrow$ SelectNghbrs($\mathcal{G}$, $q_{\text{rand}}$)
        for all $q_{\text{near}} \in Q$ do
            if Connect($q_{\text{near}}$, $q_{\text{rand}}$) then
                $\mathcal{G}$.Add($q_{\text{near}}$, $q_{\text{rand}}$)
```

```
procedure TreePlanner($q_{\text{start}}$, $q_{\text{goal}}$)
    $\mathcal{T}$.init($q_{\text{start}}$);
    while no path from $q_{\text{start}}$ to $q_{\text{goal}}$ do
        $q_{\text{rand}} \leftarrow$ Sample();
        $q_{\text{near}} \leftarrow$ Select($\mathcal{T}$, $q_{\text{rand}}$)
        $q_{\text{new}} \leftarrow$ Extend($q_{\text{near}}$, $q_{\text{rand}}$);
        if Connect($q_{\text{new}}$, $q_{\text{near}}$) then
            $\mathcal{T}$.Add($q_{\text{near}}$, $q_{\text{new}}$)
```

Figure 3.1 : Prototypical examples of graph- and tree-based sampling-based planners.

space. Most sampling-based algorithms provide probabilistic completeness guarantees [52]: if a solution exists, the probability of finding a path goes to 1 with the number of samples generated by the algorithm.. However, if no solution exists, then most sampling-based algorithms cannot recognize this (although it *is* possible [53]). Sampling-based planners fall broadly into two categories: graph-based methods such as PRM [54] and tree-based methods such as RRT [55].

3.4.1 Graph-Based Methods

Figure 3.1 shows in pseudo-code the two main varieties of sampling-based planners. Graph-based methods construct a "roadmap" within the configuration space that can be queried multiple times for motion plans. On the left is shown a basic version of the first sampling-based planner, a graph-based method known as the Probabilistic Roadmap Method (PRM) [54]. It incrementally constructs a roadmap embedded in $\mathcal{Q}_{free}$ by repeatedly sampling collision-free configurations via rejection sampling. For each sampled configuration, it computes "nearby" configurations sampled during previous iterations. If a local planner determines that there exists a collision-free motion between the new sample and a neighbor, a new edge is added to the roadmap. This process continues until the start and goal configuration are in the same connected component of the graph, at which point the shortest path in the roadmap can be extracted via, e.g., A*. The PRM algorithm grows a roadmap that can be reused for solving many motion planning problems in the same environment. First, each new start and goal configuration can be added to the roadmap. Next, the roadmap is grown (if needed) until the start and goal can be connected.

3.4.2 Tree-Based Methods

In many cases, however, solving only one particular problem is more interesting (e.g., when the environment is changing, is very large, or has many different connected components). In such cases, a tree planner as shown on the right of Figure 3.1 might be more appropriate. The most well-known variant of this type of planner is the Rapidly-exploring Random Tree (RRT) [55], but several other tree-based planners have been proposed (e.g., [56, 57, 9]). The RRT algorithm grows a tree of conformations from the start to the goal. At each iteration a random sample is generated (which may be in collision). The nearest configuration in the tree grown so far is determined and the tree is extended from this nearest configuration toward the random sample. If the new tree branch can be connected to the goal via a local planner, the algorithm terminates.

Tree-based planners can also be easily extended for kinodynamic motion planning, where the dynamics can often be written as $\dot{q} = f(q,u)$. u is a control input, a part of the robot's control space $u \in \mathcal{U}$, where $\mathcal{U}$ is a differentiable manifold. During the extension the local planner can use a steering function to drive the system towards the randomly sampled state, but even randomly sampling a control input u is generally sufficient for probabilistic completeness [55]. A popular variant of tree planners is to grow two trees simultaneously, one from the start and one from the goal [58]. After one of the trees is extended, it is checked whether the new state can be connected to nearby configurations in the other tree. The bidirectional tree search terminates once a connection between the two trees is found. Many improvements and adaptations of basic algorithms have also been proposed to handle more complex environments, such as biasing search based on previous samples to explore free space more efficiently. Both the EST [56] and KPIECE [9] planner utilize this technique.

3.4.3 Other Augmentations

The paths produced by sampling-based planning are feasible, but sometimes far from optimal. There are various techniques that post-process paths to locally optimize them [8]. This tends to work well in practice. However, with some small modifications, planners like PRM and RRT can be proven to be *asymptotically optimal*: the solution path will converge to the globally optimal solution over time [7]. Subsequent work has improved the convergence rate (see, e.g., [59]), but in practice repeatedly running a non-optimizing planner, smoothing the solution path and keeping the best one seems to work surprisingly well in comparison [60, 61]. It is also possible to create sparse roadmaps or trees that guarantee asymptotic *near*-optimality [62, 63]. That is, the solution paths converge to paths whose length is within a small constant factor approximation of the shortest path. There are complex trade-offs between the time to first feasible solution, convergence rate and degree of optimality that highly problem-dependent. Systematic benchmarking is needed to determine a good algorithm for a given problem domain [64].

Finally, an idea that can be combined with many of the planning algorithms above is lazy evaluation of the validity of configurations and the motions that connect them [65, 66]. Collision checking is the most expensive operation in sampling-based planning. By postponing this step until a candidate solution is found, collision checking can be avoided for all configurations and motions that are never considered to be part of a candidate solution.

3.4.4 Common Components

Despite their differences, sampling-based planners have similar requirements from the robot's configuration space [14]. Below are some of the components that are commonly used in sampling-based algorithms.

Samplers Sampling-based planners sample new configurations in the configuration space (`Sample` in Figure 3.1) in order to guide exploration. Typically, uniform sampling is used, but various heuristics have been proposed to sample (approximately) in lower dimensional spaces to improve the odds of sampling in narrow passages, which is key to solving the motion planning problem. Sampling near the surface of configuration space obstacles, a co-dimension 1 manifold, can be justified by the fact that configurations in narrow passages tend to be close to this surface. Although this surface is not computed analytically, various techniques have been proposed to sample *near* the surface [67, 68]. Alternatively, one could sample near the medial axis, a one-dimensional structure formed by all configurations that have more than closest point on the boundary of $\mathcal{Q}_{free}$. Configurations on the medial axis tend to "see" more of $\mathcal{Q}_{free}$ than other configurations [69, 70]. Finally, deterministic quasi-random samples have been shown to improve the dispersion compared to uniformly random sampling [71].

Metrics & nearest-neighbor data structures Usually, the natural distance metric for the configuration space is used in planning for nearest-neighbor computation, so states nearby novel states can be found (e.g., `Select` and `SelectNghbrs` in Figure 3.1). The choice of distance measure is often critical to the performance of sampling-based planners. Intuitively, a good distance measure reflects the difficulty of connecting configurations. If the measure is a proper metric, various data structures can be used to efficiently find nearest neighbors. In many cases, *approximate* neighbors are sufficient, which can be computed much more efficiently in high-dimensional spaces than exact nearest neighbors [72].

Local planner A local planner is a fast, not necessarily complete method for finding paths between nearby configurations. In sampling-based planners, geodesic movement underlies `Extend` and `Connect`, as shown in Figure 3.1. In many cases interpolation is used (or SLERP for rotations [73]). However, for kinodynamic systems a steering function could be used.

Coverage estimates Several sampling-based planners use the density of samples in a grid defined in a low-dimensional projection of the configuration space as way to measure coverage and guide the exploration [66, 74]. Although random projections often work well in practice [75], for constrained planning it may be difficult to define a projection over $\mathcal{Q}$ that approximates the density of sampling in the implicit configuration space $\mathcal{X}$.

The Open Motion Planning Library [76] provides various implementations of these core components as well as implementations of all the sampling-based planning algorithms cited in this section.

3.5 Constrained Sampling-Based Methods

Sampling-based planning with constraints introduces another element of difficulty in the problem with the need to find configurations that satisfy the constraint function. The core concepts that enable a sampling-based planner to perform effectively without the presence of constraints require adaptation to appropriately handle the constraint function, and generate a satisfying path. Let us reconsider each of the concepts introduced in the previous section, in the light of the need to satisfy the constraint function:

Samplers Sampling valid configurations is crucial to guiding the exploration of a planner through a robot's free space. However, with a constraint function that implicitly defines its valid region, the structure of this implicit region is not known *a priori*, and is thus hard to sample from without careful consideration or pre-processing. A planner needs to have the capability of either sampling valid configurations that satisfy the constraint function (finding solutions to $F(q) = \mathbf{0}$) or guiding the search in such a way so that the valid space is explored.

Metrics & nearest-neighbor data structures Normally, the distance metric utilized by a sampling-based planner is defined by the configuration space, such as the Euclidean metric for $\mathbb{R}^n$. However, the constraint function defines a subset of the configuration space that can be curved and twisted relative to the ambient configuration space. In this case, the configuration space distance metric bears very little resemblance to distance on the manifold, as is the case in, e.g., the "swiss roll" function [77]. A more appropriate metric for this space would be something like the Riemannian metric, as the constraint function defines a Riemannian manifold [12]. However, implicitly defined metrics such as the Riemannian metric are expensive to compute as it requires computation of the shortest geodesic on the manifold between two points. Computing the Riemannian metric is infeasible for any motion planning application that is concerned with speed of execution. However, if a roadmap has already been constructed on the constraint manifold then shortest path length within the roadmap could be used as an approximation of the Riemannian metric, as is done in [77]. A sampling-based planner needs to consider its choice of metric to effectively plan with constraints. Additionally, for nearest-neighbor computation, there are also approximate methods that have been employed for curved data [78]. These methods require pre-computation and are not suited for the rapidly updating structures employed in planning, and the adaptation of approximate methods is an open problem.

Local planner Normally, the local planner is a fast procedure to generate the intermediate configurations between two configurations. This, in the case of interpolation, corresponds to computing the geodesic between the configurations. However, within implicit spaces defined by constraint functions, interpolation becomes very difficult as the curvature and structure of the space is unknown *a priori*. If the constraint defines a manifold, there are many existing approaches within the literature to compute the minimum length geodesic (some not from robotics) [79, 80, 81, 82]. How a planner employs a local planner that respects constraints is crucial to its success in the constrained planning problem.

Coverage estimates To the best of the author's knowledge, no constrained sampling-based planner has employed a planning methodology that utilizes a coverage estimate in its planning process. An interesting direction for future research is to gather knowledge of the structure of the constraint manifold to direct the planning process.

For a sampling-based planner to plan with constraints, both sampling and local planning must be resolved to handle satisfying the constraint, as both of these directly affect whether the planning generates a valid path. As such, most methods focus on these two elements. For metrics, the metric from the ambient configuration space is typically used unless otherwise specified, which works well in practical applications. The ambient configuration space's metric defines a semi-metric for the constrained space, as the triangle inequality may not hold given sufficient curvature of the implicit space [83]. However, semi-metrics are "good enough" for most sampling-based planners as it only affects the effectiveness of exploration, but some theoretical guarantees may not hold.

In the following sections, the literature on constrained sampling-based planning methods is organized into high-level categories along a spectrum, which will be explained shortly. Section 3.5.1 details each of the categories, which are explained in detail in Section 3.5.2.

3.5.1 Overview

The approaches to handling constraints within a sampling-based framework can be organized into a *spectrum* which organizes them in order of the "complexity" of the algorithmic machinery necessary to compute satisfying samples and connect them to the motion graph. The spectrum is as follows, from least complex to most complex:

Relaxation As the critical limitation of sampling-based planners to solving constrained problems is their inability to find satisfying configurations (due to the lower dimension of X), a simple idea is to relax the surface of the constraint manifold by increasing the allowed tolerance of the constraint function, changing $F(q) = \mathbf{0}$ to $\|F(q)\| < \varepsilon$. With this relaxation, sampling-based planners that have no additional machinery to handle the constraint can plan and generate a path.

Projection Finding satisfying configurations of the constraint function $F(q) = \mathbf{0}$ requires finding solutions to the constraint's system of equations. From Chapter 2.3, a *projection operator* takes a configuration and *projects* it onto the surface of the implicit manifold, iteratively retracting the point to a minimum of the constraint function, solving a linear system of equations at each iteration. The projection operator is a heavy hammer available to the planner to use for both sampling and local planning.

Tangent Space From a known satisfying configuration, a *tangent space* of the constraint function can be generated (Chapter 2.3). The tangent space is constructed by finding the basis for the nullspace of the derivative of the constraint function. Satisfying configurations and valid local motions can be generated within the tangent space.

Atlas Instead of recomputing tangent spaces at all points when an expansion is needed, the tangent spaces can be kept and composed together to create a piece-wise linear *approximation* of the manifold, which can then be readily used for sampling satisfying configurations or computing geodesics for local planning. This is called an *atlas*, in a slight abuse of terminology from differential geometry.

Reparameterization For certain constraints, it is possible to compute a new parameterization of the robot's configuration, allowing direct sampling of constraint satisfying configurations. Using the reparameterized space, a new configuration space can be generated or a local motion computed and then mapped back into the robot's previous configuration space.

Each methodology will be discussed in detail in Section 3.5.2. Notably, most techniques for constrained sampling-based motion planning do not alter the core mechanics used by sampling-based planners. Generally, constrained sampling-based algorithms are adaptations of existing algorithms that incorporate a methodology for constraint satisfaction. RRT-based planners are often used as the basis for a constrained planner, perhaps in part due to the straightforward steps in the algorithm. Note that RRT-based planners rely on uniform sampling, which is typically not possible with implicit manifolds. It is an open question whether other sampling-based planners that do not depend on uniform sampling (e.g., [56, 84, 85]) might have an advantage, assuming they can be adapted to deal with constraints.

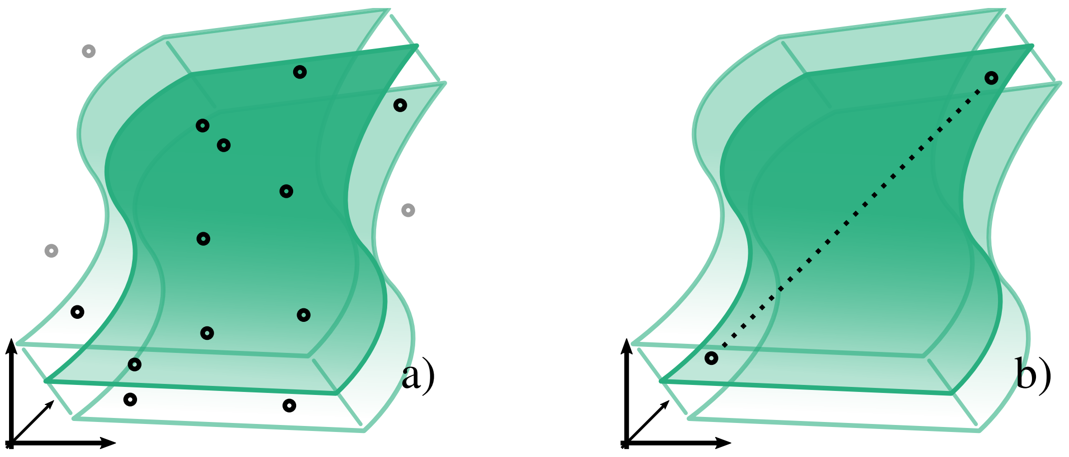


Figure 3.2 : Sampling and local planning with relaxation-based constraint handling. The constraint manifold (green) is given non-zero volume by relaxing the constraint according to some tolerance (boundaries are shown by faded extensions of the manifold). **a)** Standard rejection sampling is done to find close-to-satisfying configurations. Invalid samples are in grey, valid samples in black. **b)** Standard local planning is done (dashed line).

3.5.2 Methodologies

3.5.2.1 Relaxation Methods

Overview The primary challenge facing sampling-based planning approaches with constraints is generating configurations that satisfy the constraint equation $F(q) = \mathbf{0}$. Sampling-based planners generally sample within configuration space in which the constraint function, a set of equality relations, defines a zero volume subset. Hence there is zero probability that an uniformed random sample will satisfy the constraint. To resolve this issue, relaxation-based approaches to solving constrained problems *relax* the constraint function, growing the subset of satisfying configurations by introducing an allowable tolerance to the constraint,

$\|F(q)\| < \varepsilon$. Note that in general each of the constraint solving techniques has a tolerance on constraint satisfaction, but generally this number is tuned to achievable numerical precision to obtain accurate results. In this technique the constraint is purposefully relaxed far greater than other techniques, with ε being far greater than achievable numerical tolerance.

Literature The set of satisfying configurations has non-zero probability of being sampled within the relaxed constraint, albeit with chances similar to narrow passages in the unconstrained instance of the motion planning problem. Sampling with relaxation based approaches is depicted in Figure 3.2a. This technique is applied by [86, 87] for bi-manual manipulation problems. These works leverage execution-level controllers with compliant, closed-loop control to ensure successful execution despite configurations not within the zero-set of the constraint function. As the subset of constraint satisfying configurations defines a narrow band around the manifold, techniques well suited to motion planning problems in difficult domains can be adapted to improve performance, such as [88, 89]. Local planning is also very simple within a relaxation-based method, shown in Figure 3.2b. The local planner of the ambient configuration space is used. Since connections made to the planner's motion graph are generally local, the curvature of the manifold is respected as motions do not go far enough to invalidate themselves.

Discussion Relaxation-based approaches to constrained planning bridge unconstrained instances of the motion planning problem to the constrained incarnation. The sampling strategies employed by a relaxation-based planner can use algorithmic techniques meant to exploit lower-dimensional structures within planning such as obstacle-based sampling and medial axis sampling. As relaxation-based approaches do not fundamentally change the planning problem from the perspective of the motion planner (as they encode the constraint as a narrow passage), the constraint methodology is already decoupled from the planning

```
procedure PROJECTION($q$)
    $x \leftarrow F(q)$
    while $\|x\| > \varepsilon$ do
        $\Delta q \leftarrow J(q)^{+} x$
        $q \leftarrow q - \Delta q$
        $x \leftarrow F(q)$
    return $q$
```

An iterative procedure that uses the Jacobian pseudo-inverse ($J(q)^{+}$) of the constraint function $F(q)$. Note that the Jacobian does not need a full inversion if solutions Δq to $J(q)\Delta q = F(q)$ can be found. QR factorization [90] and other matrix decompositions might be more efficient with equivalent performance for certain problems.

Figure 3.3 : Pseudocode of a typical projection routine.

approach taken. As such, very little adaptation of any sampling-based planner is needed to handle the inflated constraint. Sampling-based planning methodologies that better suite the planning problem could be used to solve relaxed constrained planning problems. Sampling-based planners also retain their probabilistic completeness utilizing the relaxed constraint. However, execution success is no longer a given, as execution success is now determined by the controller capability to handle plans that deviate from the geometrically-defined constraint. Much of the complexity of handling the constraint is pushed onto the controller, rather than the planner. Despite these bonuses, this approach is not usually taken as it is inefficient given complex constraints. Sampling narrow passages is still inefficient compared to other approaches such as projection- or approximation-based methods. Additionally, these methods are reliant on properties of the robot and its controller, and whether the compliance of the mechanism is sufficient to retain the constraint in the face of geometric inaccuracy.

3.5.2.2 Projection Methods

Overview In a constrained motion planning problem, a satisfying path only contains configurations that satisfy the constraint function, $F(q) = \mathbf{0}$. One method to find satisfying

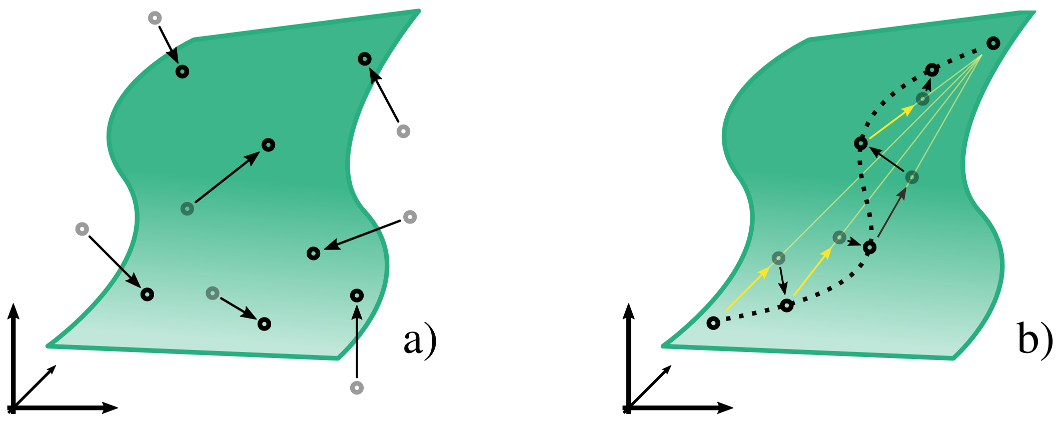


Figure 3.4 : Sampling and local planning with projection-based constraint handling. The constraint manifold (green) is projected to (black arrows) using a projection operator. **a)** After drawing a sample from configuration space (grey), it is retracted to the surface of the constraint manifold (black) using the projection operator. Local planning is shown in **b)**. From a starting configuration (bottom left), a new unsatisfying configuration (grey) closer to the goal is generated by interpolation (yellow arrow). That configuration is then projected to the manifold (black arrow), and the process continues till the goal is reached or another termination condition is met.

configurations is with a *projection operator* (as defined in Definition 2.6 in Chapter 2.3). Projection takes a configuration and *projects* it into the set of satisfying configurations, retracting the point to a minimum of the constraint function. Formally, a projection operator is an idempotent mapping $P(q) : \mathcal{Q} \to \mathcal{X}$, where if $q \in \mathcal{X}, P(q) = q$. Projection is typically an iterative optimization-based procedure that finds solutions to the constraint equation, $F(q) \approx \mathbf{0}$. A common implementation of projection is a Newton procedure with Jacobian (pseudo-)inverse gradient descent, using the Jacobian of the constraint function $J(q)$ [91, 42]. This is shown in Algorithm 1. The constraint function must encode the distance of the configuration from the solution of the equation so that the gradient adequately represents progress towards the manifold. As such, it is actually not strictly necessary that the underlying subspace defined by the constraint be a manifold, as long as this distance is properly encoded. Projection only requires piece-wise differentiability of the constraint function so that gradient descent can converge successfully.

Projection-based approaches utilize the projection operator heavily within the sampling and local planning components of the planner. Sampling with a projection operator is shown in Figure 3.4a. Samples are drawn from the ambient configuration space and are projected to solve the constraint function. As time trends to infinity, the constraint function's satisfying subset of configurations will be fully covered by projection sampling, proved in [4]. This property of projection sampling preserves the probabilistic completeness of RRT-like projection-based algorithms [4]. An example of local planning using a projection operator is shown in Figure 3.4b. In this method, the curvature of the constraint function is captured by small incremental steps interleaved with projection. From an initial satisfying configuration to a goal configuration, a small interpolation is done within the ambient configuration space. The interpolated point is projected to generate a satisfying configuration. The process is repeated from each successive point until the goal is reached. This is the core of the mechanism introduced by [29], which considers constrained motion planning for closed-loop planar chains.

Literature Historically, projection-based methods saw early adoption in solving loop-closure problems for parallel manipulators, an intrinsic constraint. Planning with loop-closures was (and continues to be) very relevant in structural biology with analytical protein analysis [92], and complex loop-closure problems in robotics were solved with PRM variants using active/passive chain methods [29, 93]. In active/passive chain methods, the projection operator uses IK to join the passive chain to the active chain, closing the loop and creating a satisfying configuration. Cyclic-Coordinate Descent [94] is another loop closure method that, unlike numerical IK, does not require the computation of Jacobian (pseudo-)inverses. Projection operators were also used to solve curve tracking problems in industrial applications early on [95].

The idea of projection to satisfy constraints was applied to general end-effector constraints in [96]. Task Constrained RRT [28] further generalized the idea of constraints and utilized Jacobian gradient descent [42] for projection. Recently, CBIRRT2 [4], the motion planner implemented for the Humanoid Path Planner System [30], and other planners such as one for the HRP2 humanoid [97] utilize projection with general constraints. Additionally, the projection methodology has been extended to handle "soft" constraints with GradienT-RRT [4]. The proposed framework can emulate CBIRRT2 and other previous approaches, as shown in Chapter 4.3.

A special application of projection-based approaches to sampling-based motion planning with constraints is the domain of *regrasping* problems. In regrasping problems, a manipulated object must be released by a manipulator (due to some obstacle within the current homotopic group of the path) and regrasped to continue progress. These problems generally define a constraint manifold which can be described as a *foliated manifold*, where the constraint manifold is divvied into slices for each pose the end-effector of the robot can take [32]. In a foliation, an n-dimensional manifold $\mathcal{X}$ is decomposed into a disjoint union of indexed, connected m-dimensional submanifolds ($m < n$) called *leaves* L_i such that $\mathcal{X} = \cup_i L_i$. Each of these leafs corresponds to the *self-motion manifold* of the robot at a particular end-effector pose, of which there are infinitely many. The self-motion manifold is the set of all configurations where the end-effector remains in the same pose, or the nullspace of the manipulator Jacobian. This property has been exploited for manipulation planning [32] and by a few constrained planners [98, 99] to achieve manipulation tasks with regrasping. Note that regrasping problems are not the exclusive domain of projection-based approaches, but have not been attempted by other types of sampling-based planners with constraints.

Discussion Projection-based planners have a number of notable advantages that contribute to their success as one of the most widely implemented methodologies to constraint handling. Primarily, the projection operator is easy to implement and captures the structure of the constraint function within the planning process. Historically, this has been implemented with randomized gradient descent in [29], but modern solvers typically implement a form of Jacobian gradient descent [28]. More advanced solvers can be used such as hierarchical inverse kinematic solvers or cyclical projection for systems under multiple constraints [4, 11]. Particularly important from an implementation perspective is the implementation of the gradient descent routine, as it requires solving a potentially complex system of equations described by the constraint at each step of the iteration. Matrix decompositions can be expensive, and the constraint Jacobian is typically not guaranteed to be invertible.

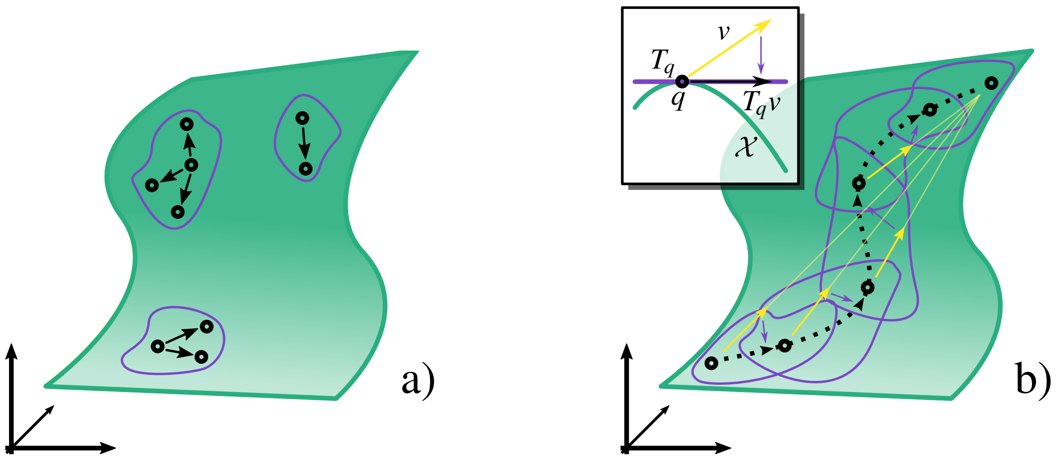


Figure 3.5 : Tangent space-based constraint handling. Given an initial satisfying configuration, a tangent space to that point can be computed which is the nullspace of the constraint function's Jacobian (purple blobs), and new satisfying configurations can be generated by local perturbations (black arrows). **a)** samples are generated by creating a tangent space at a known configuration and projecting random vectors. Local planning is shown in **b)**. From a starting configuration (bottom left), a tangent space is computed, and the vector from the current configuration to the goal is computed v (yellow arrow) and projected (purple arrow) into the tangent space (black arrowhead). The projected vector is added to the current configuration to generate a novel configuration close to satisfying the constraint.

3.5.2.3 Tangent Space Methods

Overview If constraint functions define a manifold or if the Jacobian of the constraint function is of full-rank, it is possible to locally approximate the manifold using a *tangent space* of a satisfying configuration. The tangent space defines a locally linear approximation of the constraint manifold to a Euclidean space, which extends until the curvature of the manifold bends sufficiently away. As defined in Definition 2.7 in Chapter 2.3, the tangent space is constructed by finding the basis for the nullspace of $J(q)$, which can be computed through a matrix decomposition. The tangent space T_q is a $(n-k)$-dimensional space with its origin at a configuration $q \in \mathcal{X}$, with an $n \times (n-k)$ orthonormal basis Φ_q. A vector v can also be projected through the tangent space to remove the components orthogonal to the Jacobian, leaving only components that are within the nullspace of the Jacobian, $T_q v$. This capability is primary used by algorithms based upon tangent spaces to generate new configurations. Given a satisfying configuration, the tangent space is calculated, and some random vector within the tangent space is generated (a tangent vector). The tangent vector is added to the configuration from which the tangent space was created to generate a new, local configuration that is close to the manifold. The process is depicted within Figure 3.5a. As the co-dimension of the constraint manifold approximation increases, sampling in the tangent space becomes more accurate at the price of increased computational cost per sample. Local planning utilizing tangent spaces is depicted in Figure 3.5b. From an initial configuration, a vector to the goal configuration is computed. This vector is projected into the tangent space and decomposed only into its components tangent to the manifold. The tangent vector is then added to the initial configuration to generate the next configuration in the local plan, similar to how sampling is done above. From the new configuration, the process is repeated until the goal is reached.

Literature Projection from tangent spaces was utilized within the work of [29] to generate nearby samples, which are then fixed up with projection. Tangent spaces have been used by [100, 28] for manipulators under general end-effector constraints. The technique has also seen many applications in curve tracking constraints for redundant manipulators [101, 102, 103] and structural biology to generate valid motions of proteins with loop closures [24, 104, 105].

In tangent space-based algorithms, new satisfying configurations are created by perturbing known satisfying configurations with vectors tangent to the constraint. The small perturbations create configurations that are close to satisfying the constraint, and can be projected into the satisfying set with typically few iterations. This works well for heavily constrained systems where the set of valid motions is limited. With tuning of tolerances, it is also possible to not even require reprojection of the constraint onto the manifold. Tangent space-based methods work particularly well for constraints that are closer to "linear" than curved, and are well approximated by Euclidean spaces. End-effector constraints in particular have been the target of tangent space-based methods for exploration [100, 28].

Discussion However, tangent space-based methods are not without their drawbacks. Computing the kernel of the constraint Jacobian is expensive and requires multiple matrix decompositions to solve numerically. The method also breaks down near singularity points, due to the Jacobian losing rank and no longer maintaining a surjective mapping to the ambient configuration space. Additionally, as stated above, tangent space-based methods break down when the manifold becomes highly curved, as tangent movement rapidly drifts away from the surface of the manifold.

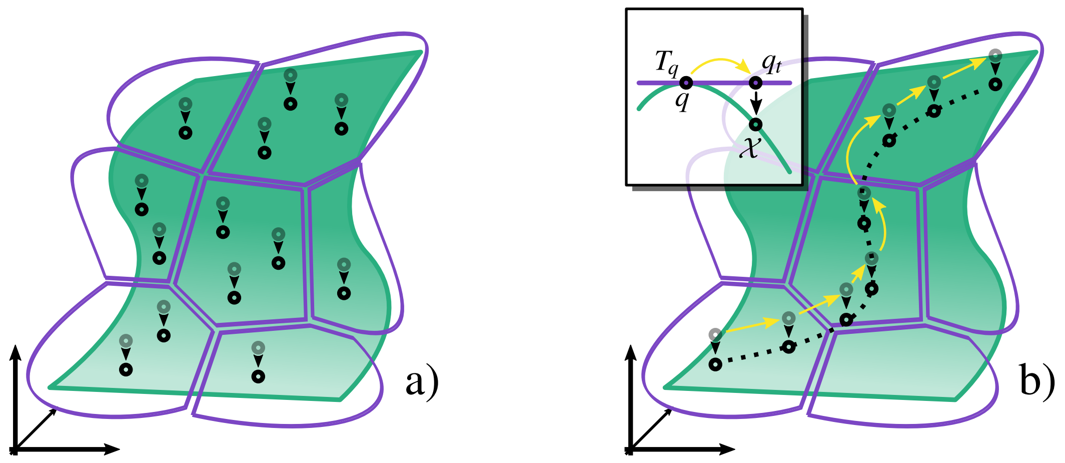


Figure 3.6 : Atlas-based sampling and local planning, akin to AtlasRRT. The atlas is a set of tangent polytopes covers the constraint manifold (purple blobs). In this figure, the atlas has already been computed and covers the space. During planning, the atlas is constructed in tandem with sampling and local planning. **a)** sampling of the manifold is done by drawing samples from the tangent polytopes by randomly sampling within (grey). These points are then orthogonally projected from polytopes to the surface of the manifold (black arrow to black), ψ_q. Local planning is shown in **b)**. Roughly, interpolation is done with the tangent space (yellow arrows) from a configuration q to another q_t, and new configurations are projected to the manifold for validation (black arrows). Reprojection is also used to switch tangent spaces (not shown). This continues until the goal is reached. See [5] for details.

3.5.2.4 Atlas Methods

Overview Furthering the idea utilizing tangent spaces to approximate the constraint locally is the idea of building an *atlas* of the manifold, a concept borrowed from the definition of differentiable manifolds [12]. Such methods also require that the constraint function defines a manifold. Unlike methods described in Section 3.5.2.3, atlas-based methods store generated tangent spaces to avoid having to compute them in a small neighborhood of the tangent space. The tangent spaces are organized within a data structure called an atlas. The atlas is defined as a piece-wise linear approximation of the constraint manifold using tangent spaces, which fully cover and approximate the manifold [106]. The key difference from the method described in [106] and atlas-based planners is the incremental construction of the atlas interweaved with space exploration. These tangent spaces are generated and utilized exactly as described above in Section 3.5.2.3, and offer capabilities of sampling and mapping to and from the manifold and the plane. Atlas-based approaches utilize the tangent spaces to project configurations to the manifold and to lift configurations into the basis defined by the tangent space.

Recall from Chapter 2.3, that a point $t \in T_q$ can be mapped into $q_t \in \mathcal{Q}$ by $q_t = q + \Phi_q t$. To map the configuration q_t onto the manifold (an exponential map ψ_q), an orthonormal projection can be computed by solving the system of equations:

$$F(q) = \mathbf{0} \qquad \text{and} \qquad \Phi_q^T(q - q_t) = \mathbf{0}$$

The opposite mapping from the manifold to T_q is much simpler:

$$\psi_q^{-1}(q_t) = t = \Phi_q^T(q - q_t)$$

. These procedures are described in detail in [13], along with other operations on implicit manifolds.

Sampling from an atlas is done as follows. A tangent space is chosen at random from the atlas, and a sample is drawn and projected from the tangent space as described in Section 3.5.2.3. This is shown in Figure 3.6a.

Literature Planners that implement variants of the atlas-based methodology are derived from the procedure described in [106]. AtlasRRT [5] implements the methodology faithfully, computing the tangent spaces and the hyperplanes to separate them (creating tangent polytopes) to guarantee uniform coverage of the part of the manifold covered by the tangent spaces. When traversing the manifold to compute connecting geodesics, AtlasRRT fully evaluates each point, projecting to the manifold orthogonally utilizing the tangent space and checking feasibility. Interpolation is done within the tangent spaces of the atlas, projecting the interpolated tangent space configuration at each step to validate the motion. This is shown in Figure 3.6b. Tangent Bundle RRT, or TB-RRT [6] is another method that utilizes atlas-based methodology. As opposed to AtlasRRT, TB-RRT performs a "lazy" evaluation and does not compute the separating halfspaces, simply collecting a set of tangent spaces that cover the manifold. TB-RRT only projects to the manifold when it needs to switch between tangent spaces, interpolating within the tangent space exclusively. Together, these features make TB-RRT more computationally efficient than AtlasRRT at exploring the constrained space. TB-RRT comes at cost of overlapping tangent spaces which leads to less uniform sampling. Futhermore, TB-RRT's lazy interpolation causes potential problems with invalid points such as failing to check collisions with narrow configuration space obstacles.

AtlasRRT has been the focus of multiple extensions, improving its capability and augmenting its guarantees. As mentioned above, one of the critical problems with generating a tangent space around a point is handling singularities, as the Jacobian of the constraint function loses rank and a tangent space can no longer be computed. In [107] a method was introduced on top of the AtlasRRT planner to plan for singularity-free paths. AtlasRRT has also been extended to be asymptotically optimal in the vein of RRT* [7] with AtlasRRT* [108]. AtlasRRT* provides the same theoretical guarantees of asymptotic optimality, while also respecting the geometric constraints imposed on the system. There has also been an extension to kinodynamic planning, or planning with non-holonomic constraints, utilizing the basic framework of AtlasRRT in [109]. Like CBIRRT2, both TB-RRT and AtlasRRT are emulated within the proposed framework, as shown in Chapter 4.4.

Discussion Atlas-based approaches make a trade-off between representational complexity and computational efficiency that pays off in many problem instances. For constraint manifolds that have complex structure and high curvature, maintaining the atlas approximation enables efficient planning regardless of the relative structure of the manifold and configuration space. For example, a constraint manifold with a toroidal topology with a narrow inner region would be hard for a projection-based approach to sample and explore due to the relatively small volume of configuration space that will end up projecting to that portion of the manifold. Atlas-based approaches would not even notice the difficulty, as they work off the constructed approximation which is invariant (given appropriate parameters) to the constraint and configuration space. This is empirically shown in Chapter 5.3. Atlas-based approaches are also probabilistically complete [5].

The primary downside to atlas-based approaches is the difficulty of implementation, as the atlas data-structure needs to be efficient and correct as it this construction is done during planning. Beyond this, there are also issues of diminishing returns with respect to the co-dimension of the constraint manifold relative to the ambient configuration space [1]. For constraints that only have a few equality constraints relative to the configuration space, maintaining an approximation of the manifold is computationally inefficient. The tangent

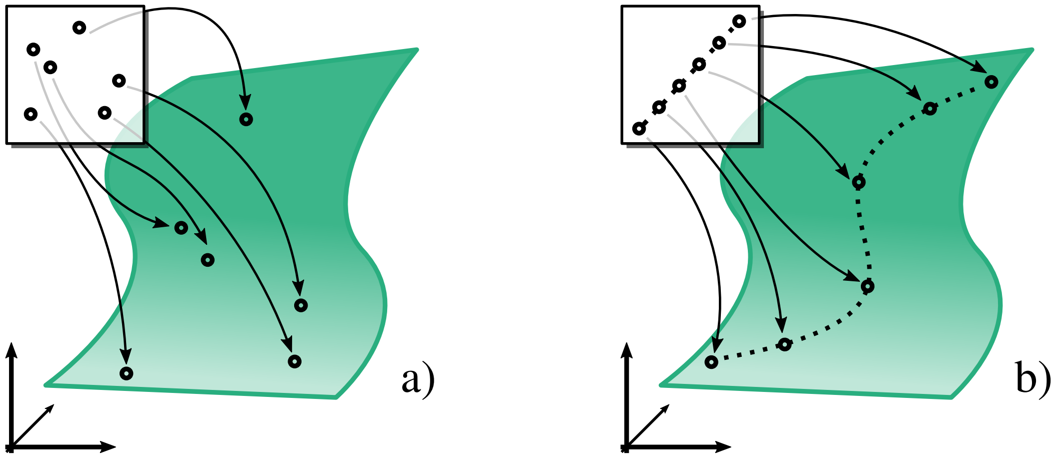


Figure 3.7 : Reparameterization-based constrained handling. A new configuration space is computed from the manipulator and constraint, reparameterizing the space (white square). **a)** samples can be draw from the reparameterized space and mapped back into the original configuration space (black arrows). Local planning is shown in **b)**. Interpolation is done within the reparameterized space and mapped back into the original configuration space.

space does not buy much over doing projection sampling in this case, as there is little difference between the constraint manifold and the ambient space.

3.5.2.5 Reparameterization Methods

Overview In some cases, the constraint function and manipulator topology lend themselves to *reparameterization*, where another configuration space is generated for the robot and constraint where each configuration fully describes the robot's state and contains only configurations that satisfy the constraint. This potentially allows for any unconstrained planner to be utilized on top of a new configuration space, enabling the machinery of the planner to function unaffected while satisfying constraints. Atlas-based approaches and others that use tangent spaces to the manifold can be thought of as reparameterization-based approaches, but a distinction made with the organization as presented here is the idea of precomputation versus online exploration and constructions. The reparameterization-based approaches presented in this subsection are usually computed before planning, while the tangent space based approaches generally are generated online for computational efficiency. Additionally, reparameterization is distinct from the tangent space and atlas approaches. Reparameterization creates a global, non-linearly-related space to the configuration space while tangent space- and atlas-based methods create local, linear approximations.

Literature Reparameterization-based methods utilize the constraint and properties of the manipulator to generate a new, reparameterized configuration space, which is then mapped back into the original configuration space. Examples of this are the deformation space for planar closed-loop systems [110] and reachable volume space [111] for general kinematic chains. Generally, these methods have their own methods for sampling within their reparameterized space (shown in Figure 3.7a), and their own methods of stepping within the reparameterized space (shown in Figure 3.7b). Deformation space reparameterizes closed-loop planar systems by encoding the "deformation" of the closed-loop within the reparameterized space. The deformation space encodes the configuration as a decomposition of triangles that form the polygon formed by the manipulator. Reachable volume space exploits properties of the joints within a kinematic chain (prismatic, revolute, and spherical) to generate *reachable volumes* of each frame of the manipulator. These are akin to Minkowski sums [3], but describe the subset of the workspace a manipulator can reach. The planner requires computing the volumes before planning, but is efficient and can scale to very large problem instances (around 70 degrees of freedom) [111].

Discussion Reparameterization-based approaches are appealing from a sampling-based planning perspective. If it was possible to simply plan within the space that contained only satisfying configurations, the constrained planning problem can be reduced to the unconstrained instance. For the unconstrained problem, this would be akin to planning only within the free configuration space. However, each of the reparameterization-based approaches requires a phase of precomputation to generate the reparameterized space. Reparameterization-based approaches heavily rely on geometrical properties of the manipulator, and use knowledge of the constraint and manipulator's shape to efficiently encode the problem. They are also generally limited to a specific model of robot (e.g., planar chains, closed loop systems) as reparameterization-based approaches require knowledge of the structure of the constraint in order to create the reparameterized space. These methods are also generally complex to implement, which prevents wide-spread applicability to many different robotic systems.

3.5.2.6 Offline Sampling

Offline sampling to solve constrained planning problems introduces a methodology orthogonal to those aforementioned. In offline sampling methods, the underlying constraint manifold is sampled before planning takes place, generating a precomputed database of constraint satisfying samples. The way these samples are generated is generally unimportant to the remainder of the planning approach, and one can utilize any of the methodologies that have been described above. Normally, projection-based approaches are used due to their ease of implementation and guarantee to cover the manifold within the limit of sampling [4]. First, samples are drawn to cover the area of interest in the satisfying subset defined by the constraint and placed within a database. Planning then takes place using standard sampling-based planning techniques, but with the planner taking its samples from the precomputed set of configurations. This approach of precomputing a set of constraint satisfying configurations was employed by [112, 113] to satisfy balancing constraints on a humanoid robot. Additionally, more structure can be imbued to the set of samples to generate a “roadmap” of valid motions on the surface of the manifold [114, 74], akin to experience-based planners for the unconstrained instance of the planning problem [115]. The self-motion manifold of a robot’s end-effector can also be precomputed utilizing “roadmap”-based methods, describing a database of inverse kinematic solutions [116].

Offline sampling-based methods have the benefit of leveraging existing techniques within the sampling-based planning literature, as they generally require minimal adaptation of a planning algorithm after the precomputed set of samples is generated. Planning is also decoupled from database generation, so the constraint sampling methodology best suited towards the particular constrained planning problem can be used. These techniques come with the obvious drawback of the need for precomputation, and the inflexibility that comes with generating a database offline. However, for intrinsic constraints of the robot, such as dynamic stability for humanoids or satisfying configurations of closed chain systems, precomputation might be the correct answer to avoid repeating computation online. Additionally, precomputation-based approaches that apply to changing environments require an element of online planning to handling changing obstacle configurations and potentially invalidates edges in an offline-computed roadmap.

3.5.3 Summary

The techniques discussed above cover a spectrum of methods to compute satisfying configurations for constrained motion planning. The spectrum describes the amount of effort the constraint methodology is using to more closely plan using the true implicit manifold. On one end of the spectrum are projection-based methods, which use little information about the constraint. On the other end lie approaches such as atlas-based methods, which compute considerable information about the constraint in order to approximate the implicit manifold.

Constrained sampling-based planning covers a large variety of methods that allow sampling-based planning algorithms to incorporate geometric motion constraints. These methods have been shown to be effective on many real-world scenarios. However, there is as of yet no consensus about which approach is best suited for which types of constraints. This likely depends on several factors: the dimensionality of the configuration space, the (co-)dimension of the constraint manifold, the degree of clutter in the environment, and so on. One factor that has not been considered in previous work is whether new exploration/exploitation strategies for planning on implicit constraint manifold are needed. As mentioned, uniform sampling and measuring distance along manifolds is either impossible or computationally very expensive. This raises the question whether a planner that depends less on uniform sampling and distance could be designed for planning with constraints.

As a first step towards unifying the approaches presented here, this thesis presents a unified framework for sampling-based planning with constraints that abstracts away the specifics of sampling and interpolating on constraint manifolds (Chapter 4). The proposed framework allows any sampling-based algorithm that is not specifically designed for constraints to plan considering constraints, using a constraint methodology presented above.

Chapter 4

The Framework

This chapter is organized as follows. First, the requirements for the proposed framework are discussed in Section 4.1. Next, the framework is discussed at an abstract level in Section 4.2 and describe how each of the space primitives utilized by a sampling-based planner are conceptualized. Then, in the following sections, emulations of three successful and widely known methodologies are shown within the framework: CBIRRT2 [4] in Section 4.3, and Tangent Bundle RRT (TB-RRT) [6] and AtlasRRT [5] in Section 4.4.

4.1 Key Components

As described in Chapter 3.4.4, sampling-based planners have similar requirements from the robot's configuration space [14]. However, as detailed in Chapter 3.5, constrained sampling-based planning introduces new challenges that must be appropriately handled when using primitive operations from the configuration space. The primary capabilities required are the following primitives previously presented in Chapter 3.4.4, detailed here with appropriate context.

Samplers Sampling "uniformly" over the space or nearby known states to generate new configurations, which can be grown towards or connected to the motion graph.

Metrics & nearest-neighbor data structures Computation of distance between states, to select nearby states in the motion graph to either extend from or connect to.

Local planner Linear interpolation on a geodesic, or moving between two states, so that new states can be created or validated through extension or connection.

Coverage estimates "Projection" for estimating configuration space coverage in relation to a task, so that the planner can measure progress and sampling can be directed towards uncovered regions (Note this is not a projection operator as described in Definition 2.6).

These can be defined as operations on the space itself and need not be specific to any planner. The contribution of this thesis is a conceptual framework, outlined within Section 4.2, that enables a broad class of motion planners to plan in many constrained spaces by exploiting the commonality of the spaces' primitive operations. This decouples constraints from a planner by augmenting the space with primitives that automatically satisfy imposed constraints.

4.2 Conceptual Framework

A sampling-based planning algorithm plans within a configuration space $\mathcal{Q}$, and generates a collision-free path by using a validity checker along with properties of the configuration space, shown in Figure 4.1a. Prior works augmented the planning algorithm with a means to find constraint satisfying motions, shown in Figure 4.1c. In contrast, the framework is a layer of abstraction that lies between the representation of the robot's configuration space and the sampling-based planner used to find valid motions, shown in Figure 4.1d. The framework can be thought of as a representation of the implicit manifold $\mathcal{X}$ defined by the constraint function F, and a means for a sampling-based planner to plan within this space.

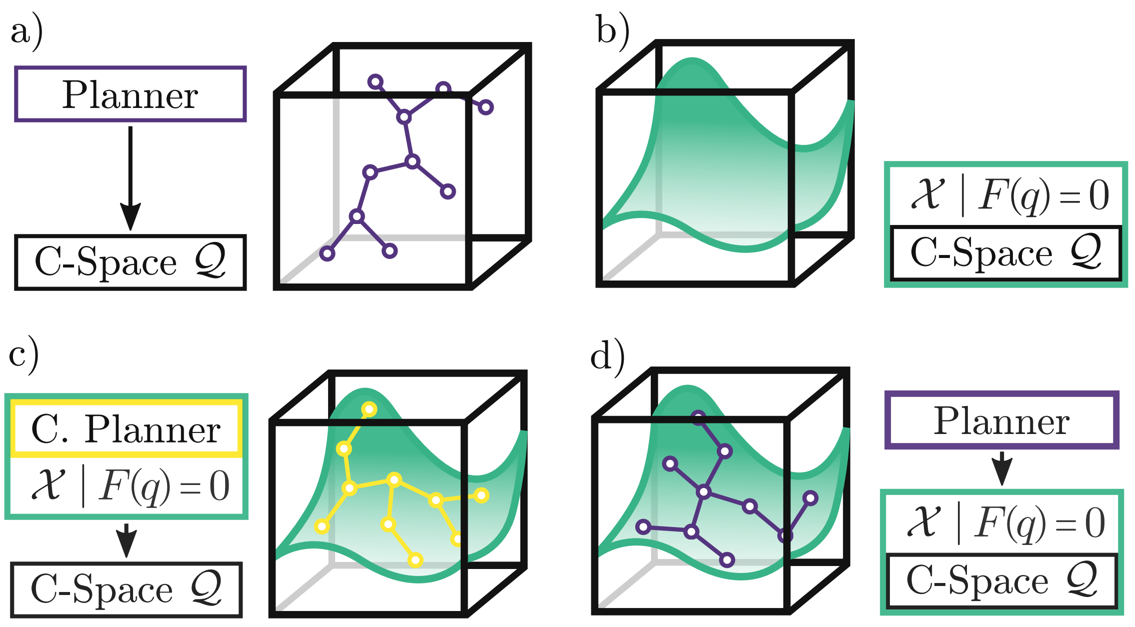


Figure 4.1 : A depiction of the framework and its relation to sampling-based planners. **a)** A box configuration space $\mathcal{Q}$ is shown in black. A sampling-based planner (purple) plans in $\mathcal{Q}$ using primitives afforded by the space. **b)** A constraint function $F(q) = \mathbf{0}$ defines a implicit manifold $\mathcal{X}$ (green). **c)** An augmented constrained sampling-based planner (yellow) (e.g., CBIRRT2, etc.) plans on $\mathcal{X}$, using its constraint methodology. **d)** The proposed framework enables any sampling-based planner (such as the unaugmented planner) to plan on $\mathcal{X}$ by incorporating $\mathcal{Q}$ and the constraint function $F(q) = \mathbf{0}$.

4.2.1 Samplers

Critical to sampling-based planners is the ability to sample new configurations in the configuration space (`Sample` in Figure 3.1). This is normally as simple as drawing uniformly random values from $\mathcal{Q}$. However, with an implicit manifold, the structure of the manifold is not known *a priori*, and is thus hard to sample uniformly without careful consideration or pre-processing. How this sampling is done is contingent on the specific constrained space, but the framework does not guarantee that it will produce uniform samples. Instead, the framework simply guarantees that any instantiation of a constraint methodology will almost-surely sample any volume of non-zero measure within the manifold.

4.2.2 Metrics & Nearest-Neighbor Data Structures

Normally, the distance metric utilized by a sampling-based planner is defined by the configuration space. This metric is primarily for nearest-neighbor computation, by which states nearby states can be found (e.g., `Select` and `SelectNghbrs` in Figure 3.1). For example, a point robot in $\mathbb{R}^3$ and a manipulator arm with $\mathcal{Q} \subseteq \mathbb{R}^n$ commonly use the Euclidean norm. However, the notion of distance in the ambient space has little meaning on the implicit manifold, as the manifold can twist and curve relative to the ambient space [77]. As the framework represents the implicit manifold of a constraint, a natural metric to use would be the Riemannian metric, or the length of the geodesic between points. However, this is computationally infeasible, as nearest-neighbor computations would require many geodesic computations. As such, the metric from the configuration space is used unless otherwise specified, defining a semi-metric on the manifold, as the triangle inequality may not hold given sufficient curvature. This is still "good enough" for most motion planning algorithms in practice, but some theoretical guarantees may not hold, such as asymptotic optimality. Note that using the ambient metric as opposed to more appropriate metrics is the state-of-the-art in regards to other constrained sampling-based planners, and a still open question.

4.2.3 Local Planner

Computing geodesics from configuration q_a to q_b normally has an analytic form, such as linear interpolation in $\mathbb{R}^n$. Geodesic movement underlies `Extend` and `Connect`, as shown

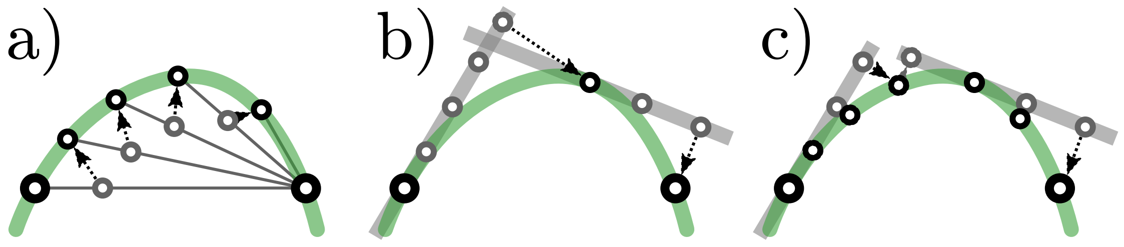


Figure 4.2 : Projection-, tangent bundle-, and atlas-based geodesic interpolation. Between points (large black) on the implicit manifold (green), the discretized geodesic is computed (black). **a)** Projection-based (CBIRRT2). Small extensions are taken (grey) and projected using a projection operator (arrow). **b)** Tangent bundle-based (TB-RRT). The manifold is lazily evaluated with tangent spaces (grey), projecting when necessary. **c)** Atlas-based (AtlasRRT). Tangent spaces are traversed, projecting at every step.

in Figure 3.1. For implicit manifolds, traversing geodesics is one of the biggest hurdles to cross. Traversing a geodesic in configuration space and attempting to “fix-up” the new configuration ignores the manifold’s curvature and can generate invalid motions. Thus, geodesic interpolation within the framework is akin to a local motion planner, computing a discretized geodesic by growing from one state to another, taking small enough steps to accurately traverse the manifold’s curvature. The way this traversal is accomplished is up to the instantiation of the methodology behind the framework, and is one of the defining traits of a constrained space. Figure 4.2 shows three local planning methodologies to compute discretized geodesics used by the three spaces in the framework. Once the discretized geodesic is computed, an interpolated state can be computed along the found geodesic, by doing piece-wise interpolation.

4.2.4 Coverage Estimates

Similar to the metric upon the space, “projection” for coverage estimates is left unaffected by the framework. As they are heuristics to bias sampling, projections are problem specific and are typically hard to devise. Random linear projections perform well in many cases, but do not incorporate constraint information [75]. Interesting future work would be to utilize information about the implicit manifold as a projection for coverage estimates.

4.2.5 Summary

In summary, the key idea of the framework is to imbue the implicit constraint manifold with primitives that closely approximate those that exist for regular configuration spaces. This allows any sampling-based planner to plan with constraints without any special consideration. The next two sections describe two approaches to sampling and interpolation that are on opposite ends of the spectrum in terms of amount of information they maintain about the constraint manifold.

4.3 Emulation of Projection-Based Methodology

One of the simplest methods to sample the constraint manifold is to sample from the configuration space and use the projection operator to retract samples onto the manifold. It was shown in [4] that sampling with projection will eventually cover the manifold, albeit with no guarantees on uniformity. Interpolation on the manifold using projection is achieved using a method similar to the extension method of the CBIRRT2 algorithm [4] (shown in Figure 4.2a). The projection-based space is emulated within the framework using the aforementioned methodology. The framework is conjectured to retain the probabilistic completeness of the overlying planners, following the proof of probabilistic completeness of projection-based RRT-like planners in [4].

4.4 Emulation of Atlas-Based Methodologies

As discussed in Chapter 2, an implicit manifold $\mathcal{X}$ can be approximated by a set of tangent spaces. A few recent planners use tangent space approximations for efficient sampling nearby the manifold, such as TB-RRT [6] and AtlasRRT [5]. The planners both sample new points by sampling within tangent spaces and projecting these points onto the manifold. Although at first biased towards explored areas, in the limit once the manifold has been fully explored sampling can approach uniform sampling [5]. These methods can sample within hard to project areas, such as the interior surface of a highly curved manifold. Geodesic interpolation is accomplished by walking along the tangent spaces of the approximation, switching tangent spaces once certain criteria are met. TB-RRT takes a lazy approach to interpolation, projecting to the manifold only when necessary to switch tangent spaces (shown in Figure 4.2b). This has the benefit of performing less work computing projections, but it is harder to do correctly. Extra consideration is needed when performing collision checking as lazy evaluation generates a relaxed geodesic, which might miss obstacles. AtlasRRT projects at every step along the approximation, and generates separating half-spaces to create polytopes of the tangent space for more accurate sampling and interpolation, at the cost of additional computation (shown in Figure 4.2c). Both of these methods are emulated within the framework.

Chapter 5

Experimental Results

This chapter presents and discusses experimental results gather using the proposed framework. The details of the framework's implementation and the experiments run are given in Section 5.1. The remaining sections detail specific experiments and their results.

5.1 Setup

The framework was implemented within the Open Motion Planning Library [76] (OMPL), which has implementations of many popular sampling-based planning algorithms. The framework fits neatly within OMPL's notion of a *state space*, and no modification was necessary to the implementation of any of the planning algorithms for them to work with the constrained planning framework. Moreover, all benchmarks were done with a single set of parameters for each constrained space and planning algorithm, to preserve fairness across multiple environments. More performance could have been gained by tuning these for each problem, but a set of reasonable defaults is desirable especially from a naïve user's perspective. All benchmarks were performed on workstations with an Intel® Core™ i7-6700K processor and 32GB of DDR4 RAM at 2400MHz.

The experiments shown here are meant to both demonstrate the effectiveness of the planning system as well as illustrate concepts that help put the work in context. The following experiments were done, and illustrate the following points:

Sphere An environment consisting of a sphere constraint within $\mathbb{R}^3$ with three obstacles.

This experiment compares the run-time of various planners and constraint methodologies. This experiment shows choice of planner (exploration strategy) can dramatically effect performance.

Torus An environment with a torus and two obstacles within $\mathbb{R}^3$. This experiment expounds upon a point made in Chapter , and shows how a problem configuration can affect the best choice of constraint methodology.

Implicit Chain An environment consisting of a spherically jointed manipulator, implicit defined by constraints. This experiment changes the number of obstacles in the scene, and shows how this effect relative performance of planners and constraint methodology.

Implicit Parallel Chain An environment with eight parallel implicit chains, creating a high-dimensional system. This experiment demonstrates the ability of the framework to scale to large problems by leveraging work in high-dimensional planning with no modification to the planner.

Additionally, some qualitative results showing asymptotically-optimal planners and other optimizing techniques in the framework is presented in Section 5.6.

5.2 Sphere Environment

Within the literature of constrained motion planning, most planners are adaptations of sampling-based planners augmented with a constraint methodology. CBIRRT2 [4], TB-RRT [6], and AtlasRRT [5], the planners emulated within the framework, all are augmentations of RRT-Connect [58]. Figure 5.1 shows the "sphere" environment, a two-dimensional

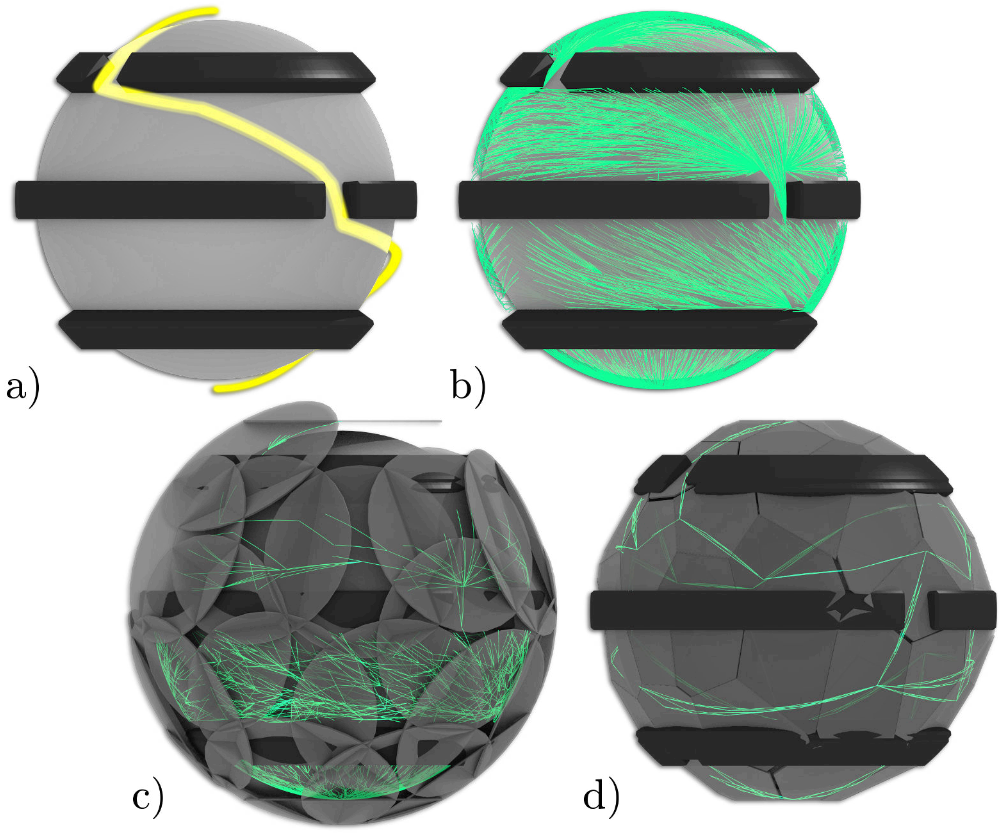


Figure 5.1 : The “sphere” environment. **a)** The sphere constraint manifold (grey) with obstacles (black). The solution path (yellow) runs from the south to north pole. **b)** Projection-based RRT* [7] motion graph (green) (Chapter 4.3). **c)** Tangent bundle-based BIT* [59] motion graph and tangent spaces (grey) (Chapter 4.4). **d)** Atlas-based SPARS [117] motion graph and tangent polytopes (Chapter 4.4).

manifold embedded within $\mathbb{R}^3$, defined by the constraint function

$$F(q) = \|q\| - 1$$

The planner must traverse three longitudinal obstacles each with a narrow passage to move from the south to the north pole. More discussion of the planners shown in Figure 5.1 is

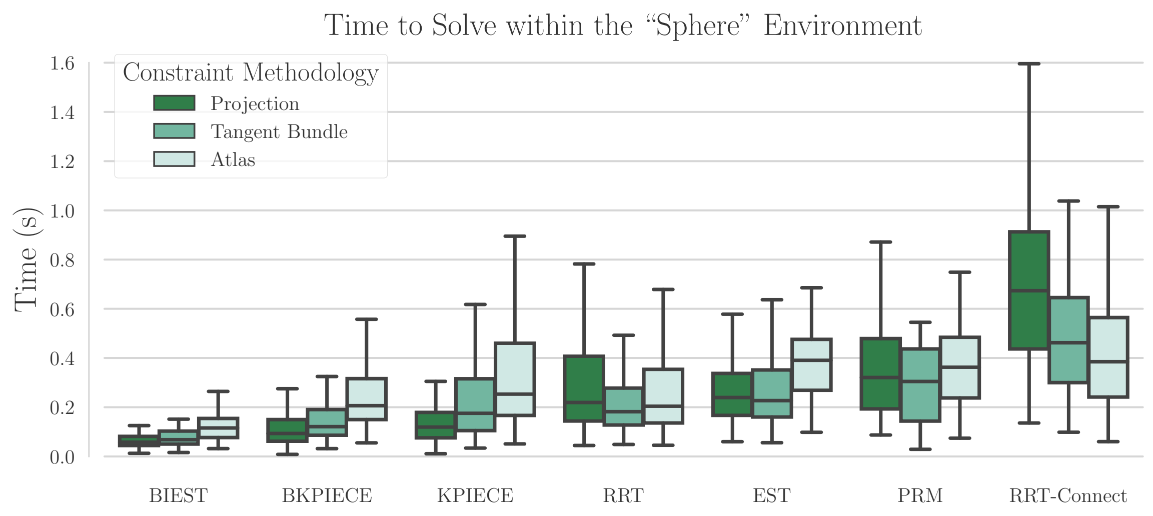


Figure 5.2 : Timing results from 100 runs of each planner in the "sphere" environment (Figure 5.1) using the three constrained spaces in the framework. Planners tested are EST, KPIECE, their bidirectional variants BIEST and BKPIECE [56, 9], RRT [55] and RRT-Connect [58], and PRM [54]. CBIRRT2, TB-RRT, and AtlasRRT are emulated by RRT-Connect in their respective constrained space, and perform the worst overall.

given in Section 5.6.

Results of 100 runs of various motion planners within the framework are shown in Figure 5.2. The planners tested are EST, KPIECE, their bidirectional variants BIEST and BKPIECE [56, 9], RRT [55] and RRT-Connect [58], and PRM [54]. A coverage estimate projection was used for KPIECE that mapped the Cartesian configuration space $(x, y, z) \in \mathbb{R}^3$ to spherical coordinates $(\theta, \phi) \in \mathbb{R}^2$,

$$\theta = \arccos(z) \qquad \text{and} \qquad \phi = \arctan\left(\frac{y}{x}\right)$$

Each of these planners were run with the three emulated constraint methodologies in the framework, the projection-, tangent bundle-, and atlas-based methodologies.

As shown in the Figure 5.2, combinations of planners and constrained spaces within the

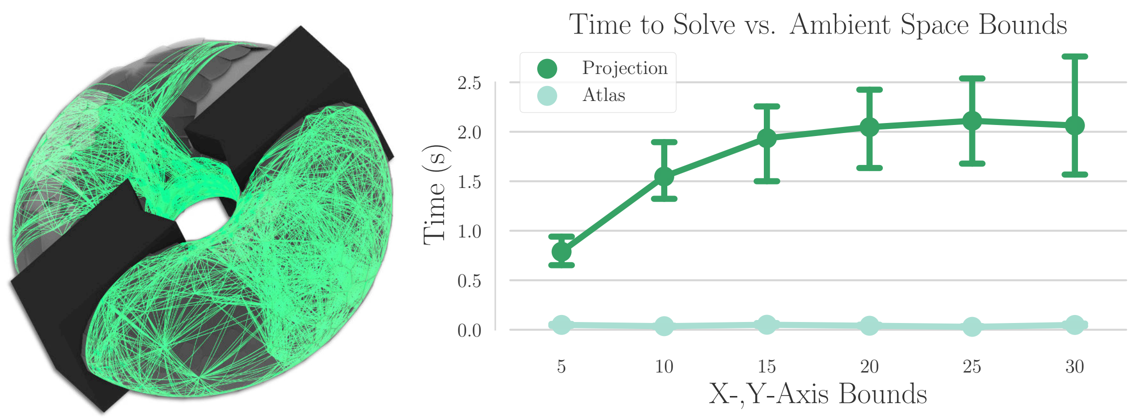


Figure 5.3 : The “torus” environment (grey) with obstacles (black) and timing results from 100 runs of PRM using the atlas- and projection-based constrained space versus the size of the *x*- and *y*- axes of the ambient configuration space. The *z*-axis bounds remained constant throughout this experiment. On the left is a PRM motion graph (green) using the atlas-based space (tangent polytopes in grey). Projection-based PRM performs orders of magnitude worse than its atlas-based counterpart.

framework have dramatically different outcomes on planning time. Previous approaches in the literature are emulated by RRT-Connect within the framework, which is shown to have the poorest performance overall within the “sphere” environment. For this problem, any of the other tested planners would be a better selection of planner if speed was the primary concern.

5.3 Torus Environment

More so, it is not just the planner that matters when approaching a constrained problem, the ambient configuration space can dramatically effect performance. Consider a “torus” environment (Figure 5.3), which is a two-dimensional manifold embedded within $\mathbb{R}^3$, with a constraint function

$$F(q) = (3 - \sqrt{x^2 + y^2})^2 + z^2 - 2.$$

The planning problem is to traverse from one end of the torus to the other. There are obstacles bound around the outer surface of the torus, allowing passage only through the inner hole to traverse from one end to the other. Timing results for the PRM planner using the projection- and atlas-based methodologies are also shown in Figure 5.3, where the total volume of the configuration space is varied while the size of the torus remains constant. As shown by the results, projection-based planning performs orders of magnitude worse than its atlas-based counterpart and worsens as the volume of the space expands, due to the inefficiency of sampling configurations that mostly project to the outer surface of the torus. The atlas-based methodology, which samples directly off of an approximation of the manifold, is unaffected by changes in the ambient configuration space. Projection to the inner surface of the torus requires sampling inside of the hole of the torus, which becomes less likely as ambient space expands. The torus example is illustrative of a problem that might arise on real robotic manipulators, as configuration spaces with revolute joints are toroidal in topology. It is unknown *a priori* how obstacles in the environment will interact with constraints, and no one constraint methodology is equipped to handle every case. Therefore, the ability to combine and change constraint methodology with a planner is essential to efficiently planning within different environments.

5.4 Implicit Chain Environment

A general trend observed is that as a planning problem becomes more constrained and the implicit manifold more curved with respect to the ambient space, atlas- and tangent-bundle-based methods perform better as the extra computation to maintain the approximation pays off. However, as the dimensionality of the problem grows, the approximation is less

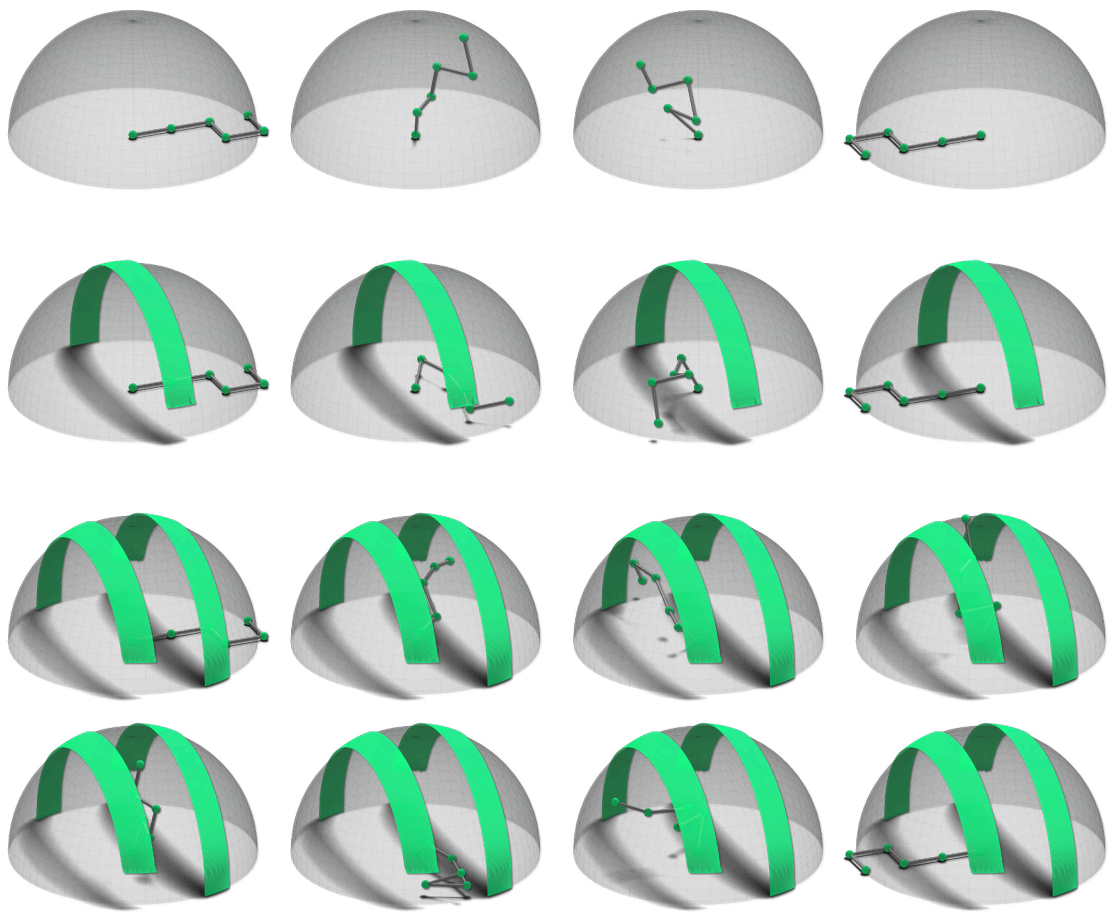


Figure 5.4 : Sample solutions paths for the no, one, and two obstacle instances of the implicit kinematic chain environment. Each of the presented paths was generated using the atlas-based constraint methodology with the KPIECE planner.

helpful and requires a similar, amortized amount of work as projection does, and projection-based methods do well. These are not rules written in stone, and there are many problems which belie their guidance. Take for example the problem of an "implicit chain", shown in Figure 5.5. Here, the kinematics are modeled as distance constraints, one for each link, on a chain with 5 spherical joints. The configuration space is thus $\mathbb{R}^{3\times 5}$. To further increase problem complexity, the following additional constraints are imposed:

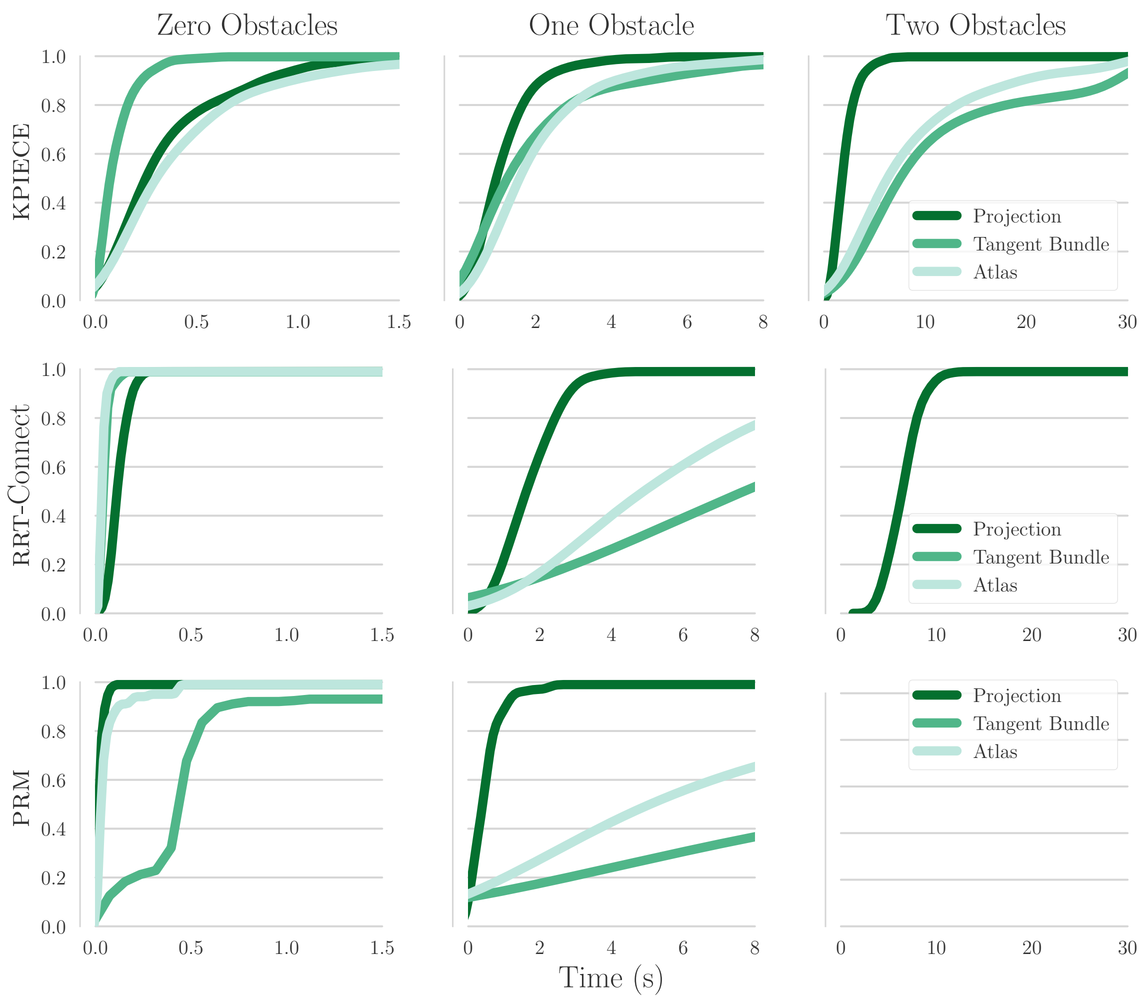


Figure 5.5 : The graphs show the cumulative probability of finding a path versus time for the KPIECE, RRT-Connect, and PRM planners using each constrained space with no surface obstacles, one obstacle, and two obstacles with antipodal narrow passages. See Figure 5.4 for sample solution paths. 100 runs were used for each cumulative probability curve, with a time-limit of 30 seconds. Note that the X-axis on each plot is different.

- The end-effector is constrained to the surface of a sphere of radius three,
- Joint 1 and 2 must have the same z-value,

- Joint 2 and 3 must have the same x-value, and

- Joint 3 and 4 must have the same z-value.

This gives an implicit manifold dimension of six. For a coverage estimate projection, a Cartesian to spherical coordinate projection was used for the end-effector's location, mapping the configuration space $(\ldots, x, y, z) \in \mathbb{R}^{15}$ to spherical coordinates upon the surface of the constraint $(\theta, \phi) \in \mathbb{R}^2$,

$$\theta = \arccos\left(\frac{z}{3}\right) \qquad \text{and} \qquad \phi = \arctan\left(\frac{y}{x}\right)$$

This captures movement of the chain across the surface of the constraint.

Timing results for this problem are shown in Figure 5.5. The following analysis is for the KPIECE planner, and results for RRT-Connect and PRM are shown as well. When there are no obstacles in this scene, tangent bundle-based methods perform the best, while projection- and atlas-based methods perform equally less. Lazy evaluation of states works in favor of this problem, as the planner can quickly traverse the constraint manifold. However, as obstacles are added to the surface of the outer sphere, tangent-bundle performs worse, as the projection and atlas methods improve relative performance drastically.

While qualitatively similar to KPIECE in many regards, one stand-out difference is the timing results for projection-based PRM with no obstacles in the scene, which handily outperforms the other two constraint methodologies. For this problem, projection from unsatisfying configurations is fairly robust and can provide reasonable answers quickly. Additionally, geodesically interpolating from distant states is easier in this environment than constructing an atlas approximation.

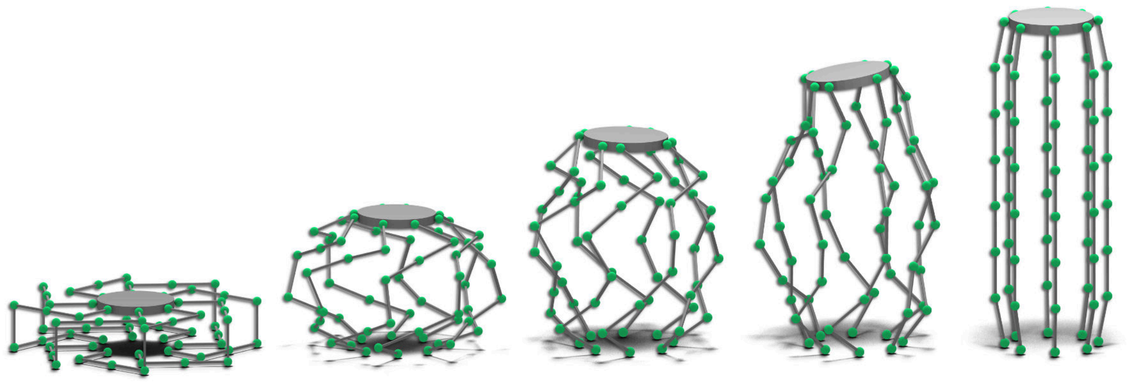


Figure 5.6 : An plan in the "implicit parallel manipulator" environment. The goal is to move from a flat, rotated configuration to upright. This path was computed using KPIECE in a projection-based constrained space in a median time of 14.5 seconds. Emulated prior approaches using RRT-Connect could not solve the problem given 10 minutes of planning time (over 100 trials).

5.5 Implicit Parallel Chain Environment

One motivating factor of this work was extending constrained planning to high-dimensional spaces, taking advantage of previous approaches in high-dimensional planning without any additional cost. Figure 5.6 shows the "implicit parallel manipulator" environment, a parallel manipulator defined with a set of the "implicit chains," defined analogously to the previous example. The end-effectors of the chains are constrained to remain attached to a shared disk, creating dependencies in their motion. The environment shown has eight chains with seven links each, for a total ambient space dimensionality of 168. The constraint manifold is of dimension 99. For a coverage estimate projection, the average height of the end-effectors was used, mapping $\mathbb{R}^{168} \to \mathbb{R}$. This captures movement of the shared disk upward, which was critical to the task at hand. Other low-dimensional projections (e.g., *x*-, *y*-, and *z*-coordinate for the centroid of the disk) performed similarly to the one-dimensional projection chosen, albeit slightly worse.

Emulated prior works (with RRT-Connect) were unable to successfully solve this system given 10 minutes of planning time. Using the KPIECE planner designed for high-dimensional spaces, the framework can quickly solve (median 14.5 seconds over 100 runs, but with high variability) this problem while satisfying constraints.

5.6 Other Results

There is little work in the literature on satisfying "soft" constraints in tandem with kinematic constraints. "Soft" constraints impose a cost over paths, and introduce a new objective in planning: finding a feasible path that minimizes the cost of the "soft" constraint. AtlasRRT* [118] and GradienT-RRT [4] both respect "soft" constraints, but require specialized implementation and integration with the constraint methodology to work. Within the proposed framework, no additional overhead is necessary for asymptotically optimal planning, as shown in Figure 5.1, which shows motion graphs for three asymptotically optimal and near-optimal planners (RRT* [7], BIT* [59], SPARS [117]). In Figure 5.1, the "soft" constraint is path length, with the optimal path being the shortest possible path from start to goal, while respecting constraints. Note that for the "soft" constraint of path length, distance was defined by the ambient configuration space, not the implicit space. Additionally, path smoothing, shortening, and interpolation algorithms work with no knowledge of constraints, as all operations are handled by the framework [8, 60, 61].

Chapter 6

Conclusion

This thesis has introduced a novel framework for constrained sampling-based planning that decouples constraint satisfaction from a motion planner's exploration of a configuration space. The framework's capability was demonstrated by showing emulations of the constraint satisfaction methodology employed by three constrained planners, CBIRRT2, TB-RRT, and AtlasRRT. Additionally, a broad range of sampling-based planners have been empirically tested within the framework for a set of constrained problems and shown that each planner can operate within the framework's constraint spaces. The framework is easily extended to new planners, and new constraint spaces can be adapted to the framework as its concepts are general to constrained planning. Although there are rough guidelines on when different constrained planning approaches tend to work better than others, for specific problems it is difficult to predict which combination of constraint space and planner will work the best. This further highlights the benefit of decoupling constraints from planning.

Currently, the framework is in the process of being integrated with real robotic platforms, as to apply the techniques seen here in simulation on physical bodies. Future work for the framework is the implementation of other constraint spaces, such as local tangent space sampling, adapting the framework for kinodynamic planning with constraints, and addressing proofs of completeness in light of the framework.

Another direction for future research is the development of a general approach to manipulation and locomotion planning that automatically identifies transitions from one set of constraints to another without requiring an hierarchical decomposition. This requires new techniques to simultaneously explore different constraint manifolds as well as ways to transition between them. A methodology to handle this is addressed in [119].

Another avenue for future work is addressing forces while planning with constraints. Intrinsic to constraints is the application of force in a specific way. For example, writing on a whiteboard is a planar geometric constraint, but also requires steady application of force to the board. The direction of force applied is orthogonal to the geometric constraint. Generally, this is translated into a geometric constraint, as in general planning with forces and dynamics is much more complicated than planning quasi-statically. Leveraging information from the constraint methodology could be potentially helpful and make force computations feasible.

Bibliography

[1] C. Voss, M. Moll, and L. E. Kavraki, "Atlas + X: Sampling-based planners on constraint manifolds," Tech. Rep. 17-02, Department of Computer Science, Rice University, Houston, TX, June 2017.

[2] J.-C. Latombe, *Robot Motion Planning*. Kluwer Academic Publishers, 1991.

[3] H. Choset, K. Lynch, S. Hutchinson, *et al.*, *Principles of Robot Motion: Theory, Algorithms, and Implementation*. MIT Press, 2005.

[4] D. Berenson, *Constrained Manipulation Planning*. PhD thesis, CMU, 2011.

[5] L. Jaillet and J. M. Porta, "Path planning under kinematic constraints by rapidly exploring manifolds," *IEEE Trans. Robot.*, vol. 29, no. 1, pp. 105–117, 2013.

[6] B. Kim, T. T. Um, C. Suh, and F. C. Park, "Tangent bundle RRT: A randomized algorithm for constrained motion planning," *Robotica*, vol. 34, no. 1, pp. 202–225, 2016.

[7] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *Int. J. of Robotics Research*, vol. 30, no. 7, pp. 846–894, 2011.

[8] R. J. Geraerts and M. H. Overmars, "Creating high-quality paths for motion planning," *Int. J. of Robotics Research*, vol. 26, no. 8, pp. 845–863, 2007.

[9] I. Şucan and L. E. Kavraki, "Kinodynamic motion planning by interior-exterior cell exploration," in *Int. Wksp. on the Algorithmic Foundations of Robotics*, 2008.

[10] Z. Kingston, M. Moll, and L. E. Kavraki, “Decoupling constraints from sampling-based planners,” in *Int. Symp. Robotics Research*, 2017. (To Appear).

[11] W. Baker, Z. Kingston, M. Moll, J. Badger, and L. E. Kavraki, “Robonaut 2 and you: Specifying and executing complex operations,” in *IEEE Wksp. on Advanced Robotics and its Social Impacts*, 2017.

[12] M. Spivak, *A Comprehensive Introduction to Differential Geometry*. Publish or Perish, 1999.

[13] W. C. Rheinboldt, “MANPAK: A set of algorithms for computations on implicitly defined manifolds,” *Computers & Mathematics with Applications*, vol. 32, no. 12, pp. 15 – 28, 1996.

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

[15] J. F. Canny, *The Complexity of Robot Motion Planning*. MIT Press, 1988.

[16] O. Khatib, “Real-time obstacle avoidance for manipulators and mobile robots,” *Int. J. of Robotics Research*, vol. 5, no. 1, pp. 90–98, 1986.

[17] J. Barraquand and J.-C. Latombe, “Nonholonomic multibody mobile robots: Controllability and motion planning in the presence of obstacles,” *Algorithmica*, vol. 10, no. 2, pp. 121–155, 1993.

[18] M. Phillips, V. Hwang, S. Chitta, and M. Likhachev, “Learning to plan for constrained manipulation from demonstrations,” *Autonomous Robots*, vol. 40, no. 1, pp. 109–124, 2016.

[19] J.-C. Latombe, “Motion planning: A journey of robots, molecules, digital actors, and other artifacts,” *Int. J. of Robotics Research*, vol. 18, no. 11, pp. 1119–1128, 1999.

[20] B. Gipson, D. Hsu, L. E. Kavraki, and J.-C. Latombe, “Computational models of protein kinematics and dynamics: Beyond simulation,” *Annual Review of Analytical Chemistry*, vol. 5, pp. 273–291, 2012.

[21] A. P. Ambler and R. J. Popplestone, “Inferring the positions of bodies from specified spatial relationships,” *Artificial Intelligence*, vol. 6, no. 2, pp. 157–174, 1975.

[22] M. T. Mason, “Compliance and force control for computer controlled manipulators,” *IEEE Trans. on Syst., Man, and Cybernetics*, 1981.

[23] O. Khatib, “A unified approach for motion and force control of robot manipulators: The operational space formulation,” *IEEE J. Robot. Autom.*, vol. 3, no. 1, pp. 43–53, 1987.

[24] Y. Zhang and K. Hauser, “Unbiased, scalable sampling of protein loop conformations from probabilistic priors,” *BMC Structural Biology*, vol. 13, no. 1, p. S9, 2013.

[25] J. S. B. Mitchell, D. M. Mount, and C. H. Papdimitrious, “The discrete geodesic problem,” *SIAM J. on Computing*, vol. 16, no. 4, pp. 647–668, 1987.

[26] T. Asano, T. Asano, L. Guibas, J. Hershberger, and H. Imai, “Visibility-polygon search and Euclidean shortest paths,” in *Annual Symp. on Foundations of Computer Science*, pp. 155–164, 1985.

[27] C. Alexopoulos and P. M. Griffin, “Path planning for a mobile robot,” *IEEE Trans. on Syst., Man, and Cybernetics*, vol. 22, no. 2, pp. 318–322, 1992.

[28] M. Stilman, “Global manipulation planning in robot joint space with task constraints,” *IEEE Trans. Robot.*, vol. 26, no. 3, pp. 576–584, 2010.

[29] J. H. Yakey, S. M. LaValle, and L. E. Kavraki, "Randomized path planning for linkages with closed kinematic chains," *IEEE Trans. Robot. Autom.*, vol. 17, no. 6, pp. 951–958, 2001.

[30] J. Mirabel, S. Tonneau, P. Fernbach, *et al.*, "HPP: A new software for constrained motion planning," in *IEEE/RSJ Int. Conf. on Intell. Robots and Syst.*, pp. 383–389, 2016.

[31] L. Sentis and O. Khatib, "Synthesis of whole-body behaviors through hierarchical control of behavioral primitives," *Int. J. of Humanoid Robotics*, vol. 2, no. 4, pp. 505–518, 2005.

[32] T. Siméon, J.-C. Laumond, J. Cortés, and A. Sahbani, "Manipulation planning with probabilistic roadmaps," *Int. J. of Robotics Research*, vol. 32, no. 7-8, pp. 729–746, 2004.

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

[34] J. Mirabel and F. Lamiraux, "Manipulation planning: addressing the crossed foliation issue," in *IEEE Int. Conf. Robot. Autom.*, 2017.

[35] 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 *IEEE Int. Conf. Robot. Autom.*, 2012.

[36] W. Reid, R. Fitch, A. H. Göktogan, and S. Sukkarieh, "Motion planning for reconfigurable mobile robots using hierarchical fast marching trees," in *Int. Wksp. on the Algorithmic Foundations of Robotics*, 2016.

[37] N. T. Dantam, Z. Kingston, S. Chaudhuri, and L. E. Kavraki, "Incremental task and motion planning: A constraint-based approach," in *Robotics: Science and Syst.*, 2016.

[38] T. Lozano-Pérez and L. P. Kaelbling, "A constraint-based method for solving sequential manipulation planning problems," in *IEEE/RSJ Int. Conf. on Intell. Robots and Syst.*, 2014.

[39] L. Kaelbling and T. Lozano-Pérez, "Hierarchical task and motion planning in the now," in *IEEE Int. Conf. Robot. Autom.*, pp. 1470–1477, 2011.

[40] S. Seereeram and J. T. Wen, "A global approach to path planning for redundant manipulators," *IEEE Trans. Robot. Autom.*, vol. 11, no. 1, pp. 152–160, 1995.

[41] D. E. Whitney, "Resolved motion rate control of manipulators and human prostheses," *IEEE Trans. on Man-Machine Syst.*, vol. 10, no. 2, pp. 47–53, 1969.

[42] S. R. Buss and J.-S. Kim, "Selectively damped least squares for inverse kinematics," *J. of Graphics, GPU, and Game Tools*, vol. 10, no. 3, pp. 37–49, 2005.

[43] J. James, Y. Weng, S. Hart, *et al.*, "Prophetic goal-space planning for human-in-the-loop mobile manipulation," in *IEEE-RAS Int. Conf. on Humanoid Robots*, pp. 1185–1192, 2015.

[44] M. Fallon, S. Kuindersma, S. Karumanchi, *et al.*, "An architecture for online affordance-based perception and whole-body planning," *J. of Field Robotics*, vol. 32, no. 2, pp. 229–254, 2015.

[45] M. Johnson, B. Shrewsbury, S. Bertrand, T. Wu, D. Duran, *et al.*, "Team IHMC's lessons learned from the DARPA robotics challenge trials," *J. of Field Robotics*, vol. 32, no. 2, pp. 192–208, 2015.

[46] C. Atkeson, N. Banerjee, D. Berenson, M. DeDonato, R. Du, *et al.*, “NO FALLS, NO RESETS: Reliable humanoid behavior in the DARPA robotics challenge,” in *Int. Wksp. on the Algorithmic Foundations of Robotics*, pp. 623–630, 2015.

[47] M. Zucker, N. Ratliff, A. Dragan, *et al.*, “CHOMP: Covariant hamiltonian optimization for motion planning,” *Int. J. of Robotics Research*, 2013.

[48] M. Kalakrishnan, S. Chitta, E. Theodorou, P. Pastor, and S. Schaal, “STOMP: Stochastic trajectory optimization for motion planning,” in *IEEE Int. Conf. Robot. Autom.*, pp. 4569–4574, 2011.

[49] J. Schulman, Y. Duan, J. Ho, *et al.*, “Motion planning with sequential convex optimization and convex collision checking,” *Int. J. of Robotics Research*, vol. 33, no. 9, pp. 1251–1270, 2014.

[50] J. Dong, M. Mukadam, F. Dellaert, and B. Boots, “Motion planning as probabilistic inference using Gaussian processes and factor graphs,” in *Robotics: Science and Syst.*, 2016.

[51] M. Elbanhawi and M. Simic, “Sampling-based robot motion planning: A review,” *IEEE Access*, vol. 2, pp. 56–77, 2014.

[52] A. M. Ladd and L. E. Kavraki, “Measure theoretic analysis of probabilistic path planning,” *IEEE Trans. Robot. Autom.*, vol. 20, no. 2, pp. 229–242, 2004.

[53] Z. McCarthy, T. Bretl, and S. Hutchinson, “Proving path non-existence using sampling and alpha shapes,” in *IEEE Int. Conf. Robot. Autom.*, pp. 2563–2569, 2012.

[54] L. E. Kavraki, P. Švestka, J.-C. Latombe, and M. Overmars, “Probabilistic roadmaps for path planning in high-dimensional configuration spaces,” *IEEE Trans. Robot.*

Autom., vol. 12, no. 4, pp. 566–580, 1996.

[55] S. M. LaValle and J. J. Kuffner, "Randomized kinodynamic planning," *Int. J. of Robotics Research*, vol. 20, no. 5, pp. 378–400, 2001.

[56] D. Hsu, J.-C. Latombe, and R. Motwani, "Path planning in expansive configuration spaces," *Int. J. of Computational Geometry and Applications*, vol. 9, no. 4n5, pp. 495–512, 1999.

[57] A. M. Ladd and L. E. Kavraki, "Fast tree-based exploration of state space for robots with dynamics," in *Algorithmic Foundations of Robotics VI* (M. Erdmann, D. Hsu, M. Overmars, and A. F. van der Stappen, eds.), pp. 297–312, Springer, STAR 17, 2005.

[58] J. J. Kuffner and S. M. LaValle, "RRT-Connect: An efficient approach to single-query path planning," in *IEEE Int. Conf. Robot. Autom.*, vol. 2, pp. 995–1001, 2000.

[59] J. Gammell, S. S. Srinivasa, and T. D. Barfoot, "Batch informed trees (BIT*): Sampling-based optimal planning via the heuristically guided search of implicit random geometric graphs," in *IEEE Int. Conf. Robot. Autom.*, 2015.

[60] R. Luna, I. A. Şucan, M. Moll, and L. E. Kavraki, "Anytime solution optimization for sampling-based motion planning," in *IEEE Int. Conf. Robot. Autom.*, 2013.

[61] J. Luo and K. Hauser, "An empirical study of optimal motion planning," in *IEEE/RSJ Int. Conf. on Intell. Robots and Syst.*, 2014.

[62] A. Dobson and K. E. Bekris, "Sparse roadmap spanners for asymptotically near-optimal motion planning," *Int. J. of Robotics Research*, vol. 33, no. 1, pp. 18–47, 2014.

[63] Y. Li, Z. Littlefield, and K. E. Bekris, "Asymptotically optimal sampling-based kinodynamic planning," *Int. J. of Robotics Research*, vol. 35, pp. 528–564, 2016.

[64] M. Moll, I. A. Şucan, and L. E. Kavraki, "Benchmarking motion planning algorithms: An extensible infrastructure for analysis and visualization," *IEEE Robot. & Autom. Magazine*, vol. 22, no. 3, pp. 96–102, 2015.

[65] R. Bohlin and L. E. Kavraki, "Path planning using lazy PRM," in *IEEE Int. Conf. Robot. Autom.*, pp. 521–528, 2000.

[66] G. Sánchez and J.-C. Latombe, "A single-query bi-directional probabilistic roadmap planner with lazy collision checking," in *Int. Symp. on Robotics Research*, pp. 403–417, 2001.

[67] N. Amato, O. Bayazit, L. Dale, C. Jones, and D. Vallejo, "OBPRM: An obstacle-based PRM for 3D workspaces," in *Robotics: The Algorithmic Perspective* (P. K. Agarwal, L. E. Kavraki, and M. T. Mason, eds.), pp. 155–168, A.K. Peters, 1999.

[68] V. Boor, M. H. Overmars, and A. F. van der Stappen, "The Gaussian sampling strategy for probabilistic roadmap planners," in *IEEE Int. Conf. Robot. Autom.*, pp. 1018–1023, 1999.

[69] S. A. Wilmarth, N. Amato, and P. F. Stiller, "MAPRM: A probabilistic roadmap planner with sampling on the medial axis of the free space," in *IEEE Int. Conf. Robot. Autom.*, (Detroit, MI), pp. 1024–1031, May 1999.

[70] D. Hsu, T. Jiang, J. Reif, and Z. Sun, "The bridge test for sampling narrow passages with probabilistic roadmap planners," *IEEE Int. Conf. Robot. Autom.*, vol. 3, pp. 4420–4426, 2003.

[71] S. M. LaValle, M. S. Branicky, and S. R. Lindemann, "On the relationship between classical grid search and probabilistic roadmaps," *Int. J. of Robotics Research*, vol. 23, no. 7-8, p. 673, 2004.

[72] A. Andoni and P. Indyk, "Nearest neighbours in high-dimensional spaces," in *Handbook of Discrete and Computational Geometry* (J. E. Goodman, J. O'Rourke, and C. D. Tóth, eds.), ch. 43, CRC Press, third ed., 2017.

[73] K. Shoemake, "Animating rotation with quaternion curves," in *ACM SIGGRAPH Computer Graphics*, vol. 19, pp. 245–254, 1985.

[74] I. A. Şucan and S. Chitta, "Motion planning with constraints using configuration space approximations," *IEEE/RSJ Intl. Conf. on Intell. Robots and Syst.*, 2012.

[75] I. A. Şucan and L. E. Kavraki, "On the performance of random linear projections for sampling-based motion planning," in *IEEE/RSJ Int. Conf. on Intell. Robots and Syst.*, 2009.

[76] I. Şucan, M. Moll, and L. E. Kavraki, "The Open Motion Planning Library," *IEEE Robot. Autom. Magazine*, vol. 19, no. 4, pp. 72–82, 2012.

[77] J. B. Tenenbaum, V. de Silva, and J. C. Langford, "A global geometric framework for nonlinear dimensionality reduction," *Science*, 2000.

[78] R. Chaudhry and Y. Ivanov, "Fast approximate nearest neighbor methods for non-Euclidean manifolds with applications to human activity analysis in videos," in *European Conf. on Comput. Vision*, pp. 735–748, Springer, 2010.

[79] R. Kimmel and J. A. Sethian, "Computing geodesic paths on manifolds," *Proc. National Academy of Sciences*, vol. 95, no. 15, pp. 8431–8435, 1998.

[80] L. Ying and E. J. Candes, “Fast geodesic computation with the phase flow method,” *J. of Computational Physics*, vol. 220, no. 1, pp. 6–18, 2006.

[81] K. Crane, C. Weischedel, and M. Wardetzky, “Geodesics in heat: A new approach to computing distance based on heat flow,” *ACM Trans. on Graphics*, vol. 32, pp. 152:1–152:11, Oct. 2013.

[82] K. Hauser, “Fast interpolation and time-optimization with contact,” *Int. J. of Robotics Research*, vol. 33, no. 9, pp. 1231–1250, 2014.

[83] M. M. Deza and E. Deza, *Encyclopedia of Distances*. Springer, 2009.

[84] A. M. Ladd and L. E. Kavraki, “Motion planning in the presence of drift, underactuation and discrete system changes,” in *Robotics: Science and Systems*, (Boston, MA), pp. 233–241, MIT Press, June 2005.

[85] I. A. Şucan and L. E. Kavraki, “A sampling-based tree planner for systems with complex dynamics,” *IEEE Trans. Robot.*, vol. 28, no. 1, pp. 116–131, 2012.

[86] M. Bonilla, E. Farnioli, L. Pallottino, and A. Bicchi, “Sample-based motion planning for soft robot manipulators under task constraints,” in *IEEE Int. Conf. Robot. Autom.*, pp. 2522–2527, 2015.

[87] M. Bonilla, L. Pallottino, and A. Bicchi, “Noninteracting constrained motion planning and control for robot manipulators,” in *IEEE Int. Conf. Robot. Autom.*, 2017.

[88] S. Rodriguez, S. Thomas, R. Pearce, and N. M. Amato, “RESAMPL: A region-sensitive adaptive motion planner,” *Algorithmic Foundation of Robotics VII*, pp. 285–300, 2008.

[89] J. Bialkowski, M. Otte, and E. Frazzoli, "Free-configuration biased sampling for motion planning," in *IEEE/RSJ Int. Conf. on Intell. Robots and Syst.*, pp. 1272–1279, 2013.

[90] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery, *Numerical Recipes in C: The art of Scientific Computing*. Cambridge University Press, second ed., 1992.

[91] J. Dennis and R. Schnabel, *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*. Englewood Cliffs, NJ: Prentice-Hall, 1983.

[92] W. J. Wedemeyer and H. A. Scheraga, "Exact analytical loop closure in proteins using polynomial equations," *J. of Computational Chemistry*, vol. 20, no. 8, pp. 819–844, 1999.

[93] J. Cortés, T. Siméon, and J.-P. Laumond, "A random loop generator for planning the motions of closed kinematic chains using PRM methods," in *IEEE Int. Conf. Robot. Autom.*, vol. 2, pp. 2141–2146, 2002.

[94] A. A. Canutescu and R. L. Dunbrack, "Cyclic coordinate descent: A robotics algorithm for protein loop closure," *Protein Science*, vol. 12, no. 5, pp. 963–972, 2003.

[95] G. Oriolo, M. Ottavi, and M. Vendittelli, "Probabilistic motion planning for redundant robots along given end-effector paths," in *IEEE/RSJ Int. Conf. on Intell. Robots and Syst.*, vol. 2, pp. 1657–1662, 2002.

[96] Z. Yao and K. Gupta, "Path planning with general end-effector constraints: Using task space to guide configuration space search," in *IEEE/RSJ Int. Conf. on Intell. Robots and Syst.*, pp. 1875–1880, 2005.

[97] F. Kanehiro, E. Yoshida, and K. Yokoi, "Efficient reaching motion planning and execution for exploration by humanoid robots," in *IEEE/RSJ Int. Conf. on Intell. Robots and Syst.*, pp. 1911–1916, 2012.

[98] J. Kim, I. Ko, and F. C. Park, "Randomized path planning for tasks requiring the release and regrasp of objects," *Advanced Robotics*, vol. 30, no. 4, pp. 270–283, 2016.

[99] Z. Xian, P. Lertkultanon, and Q. C. Pham, "Closed-chain manipulation of large objects by multi-arm robotic systems," *IEEE Robot. Autom. Letters*, vol. 2, no. 4, pp. 1832–1839, 2017.

[100] M. V. Weghe, D. Ferguson, and S. S. Srinivasa, "Randomized path planning for redundant manipulators without inverse kinematics," in *IEEE-RAS Int. Conf. on Humanoid Robots*, pp. 477–482, 2007.

[101] G. Oriolo and M. Vendittelli, "A control-based approach to task-constrained motion planning," in *IEEE/RSJ Int. Conf. on Intell. Robots and Syst.*, pp. 297–302, 2009.

[102] M. Vendittelli and G. Oriolo, "Task-constrained motion planning for underactuated robots," in *IEEE Int. Conf. Robot. Autom.*, pp. 2965–2970, 2009.

[103] M. Cefalo, G. Oriolo, and M. Vendittelli, "Task-constrained motion planning with moving obstacles," in *IEEE/RSJ Int. Conf. on Intell. Robots and Syst.*, pp. 5758–5763, 2013.

[104] D. V. Pachov and H. van den Bedem, "Nullspace sampling with holonomic constraints reveals molecular mechanisms of protein Gαs," *PLoS Computational Biology*, vol. 11, no. 7, 2015.

[105] R. Fonseca, D. Budday, and H. van dem Bedem, "Collision-free poisson motion planning in ultra high-dimensional molecular conformation spaces," *arXiv preprint*, 2016.

[106] M. E. Henderson, "Multiple parameter continuation: Computing implicitly defined k-manifolds," *Int. J. of Bifurcation and Chaos*, vol. 12, no. 3, pp. 451–476, 2002.

[107] O. Bohigas, M. E. Henderson, L. Ros, M. Manubens, and J. M. Porta, "Planning singularity-free paths on closed-chain manipulators," *IEEE Trans. Robot.*, vol. 29, no. 4, pp. 888–898, 2013.

[108] L. Jaillet and J. M. Porta, "Asymptotically-optimal path planning on manifolds," *Robotics: Science and Syst.*, 2013.

[109] R. Bordalba, L. Ros, and J. M. Porta, "Kinodynamic planning on constraint manifolds," *ArXiv e-prints*, 2017.

[110] L. Han, L. Rudolph, J. Blumenthal, and I. Valodzin, "Convexly stratified deformation spaces and efficient path planning for planar closed chains with revolute joints," *Int. J. of Robotics Research*, vol. 27, no. 11-12, pp. 1189–1212, 2008.

[111] T. McMahon, *Sampling Based Motion Planning with Reachable Volumes*. PhD thesis, Texas A&M University, 2016.

[112] J. J. Kuffner, S. Kagami, K. Nishiwaki, M. Inaba, and H. Inoue, "Dynamically-stable motion planning for humanoid robots," *Autonomous Robots*, vol. 12, no. 1, pp. 105–118, 2002.

[113] F. Burget, A. Hornung, and M. Bennewitz, "Whole-body motion planning for manipulation of articulated objects," in *IEEE Int. Conf. Robot. Autom.*, pp. 1656–1662,

2013.

[114] T. Igarashi and M. Stilman, “Homotopic path planning on manifolds for cabled mobile robots,” in *Int. Wksp. on the Algorithmic Foundations of Robotics*, pp. 1–18, 2010.

[115] D. Coleman, I. A. Şucan, M. Moll, K. Okada, and N. Correll, “Experience-based planning with sparse roadmap spanners,” in *IEEE Int. Conf. Robot. Autom.*, pp. 900–905, 2015.

[116] K. Hauser, “Continuous pseudoinversion of a multivariate function: Application to global redundancy resolution,” *Int. Wksp. on the Algorithmic Foundations of Robotics*, 2016.

[117] A. Dobson, A. Krontiris, and K. E. Bekris, “Sparse roadmap spanners,” in *Int. Wksp. on the Algorithmic Foundations of Robotics*, 2013.

[118] L. Jaillet and J. M. Porta, “Efficient asymptotically-optimal path planning on manifolds,” *Robotics and Autonomous Syst.*, vol. 61, no. 8, pp. 797–807, 2013.

[119] M. E. Henderson, “Multiparameter parallel search branch switching,” *Int J. of Bifurcation and Chaos*, vol. 15, no. 03, pp. 967–974, 2005.