Accepted for publication in *Annual Review of Control, Robotics, and Autonomous Systems,* 2018
http://www.annualreviews.org/journal/control

# Sampling-Based Methods for Motion Planning with Constraints

**Zachary Kingston,**[1] **Mark Moll,**[1]
**and Lydia E. Kavraki**[1]

[1] Department of Computer Science, Rice University, Houston, Texas, 77005; email: {zak, mmoll, kavraki}@rice.edu

Xxxx. Xxx. Xxx. Xxx. YYYY. AA:1–31

https://doi.org/10.1146/((please add article doi))



**Keywords**

robotics, robot motion planning, sampling-based planning, constraints, planning with constraints, planning for high-dimensional robotic systems

**Abstract**

Robots with many degrees of freedom (e.g., humanoid robots and mobile manipulators) have increasingly been employed to accomplish realistic tasks in domains such as disaster relief, spacecraft logistics, and home caretaking. Finding feasible motions for these robots autonomously is essential for their operation. Sampling-based motion planning algorithms have been shown to be effective for these high-dimensional systems. However, incorporating *task constraints* (e.g., keeping a cup level, writing on a board) into the planning process introduces significant challenges. This survey describes the families of methods for sampling-based planning with constraints and places them on a spectrum delineated by their complexity. Constrained sampling-based methods are based upon two core primitive operations: (1) sampling constraint-satisfying configurations and (2) generating constraint-satisfying continuous motion. Although the basics of sampling-based planning are presented for contextual background, the survey focuses on the representation of constraints and sampling-based planners that incorporate constraints.

## Contents



## 1. INTRODUCTION

Consider a mobile manipulator, tasked with carrying out various chores and maintenance tasks. Robonaut 2 (R2) (1), shown in Figure 1, is an example of such a high-dimensional robotic platform, designed to work with humans in spacecraft. Future concept NASA missions involve spacecraft to be uncrewed for large periods of time, but the spacecraft will still require maintenance. R2, and other robots, could fill a unique role in such missions and are mechanically capable of executing the complex tasks necessary for spacecraft upkeep, such as turning valves (Figure 1a), opening doors, and extracting cargo from a rack (Figure 1b). However, for these robots to do so autonomously, a *motion planning* system that can generate feasible motion from high-level requirements is needed. Additionally, this motion planning system must respect the *constraints* on a task in order achieve success, such as turning the valve only about its axis, rotating the door about its hinge, or extracting the cargo linearly from its hold. The requirements produced by this scenario and in many other problem domains (e.g., household caretaking, disaster recovery) motivate the study of sampling-based motion planners with constraints.

Motion planning is an essential tool for autonomous robots, as it enables description and execution of motion as high-level goals, rather than lower-level primitives (e.g., manual specification of joint angles for a manipulator). However, motion planning is a PSPACE-hard problem (2), with complexity growing with the number of a robot's degrees of freedom. Although exact algorithms exist, they are difficult to implement and scale poorly to high-dimensional

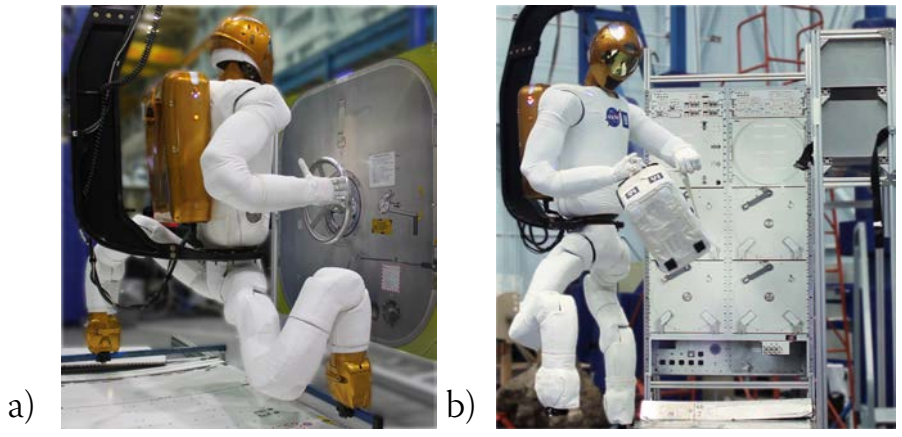



**Figure 1**

Examples of motion planning with constraints. **a)** NASA's Robonaut 2, a highly dextrous humanoid robot, opens a valve, which constrains the end effector to follow a circular motion. Next, it slides open the door, which requires a whole-body sideways movement while maintaining its stance. **b)** Robonaut2 holds a bag with two hands and move its entire body. Credit: NASA.

robots. Before going into detail on constrained sampling-based planning algorithms, we first provide a brief summary of historical approaches to the motion planning problem and point out which approaches have considered constraints.

Early on, potential field methods were proposed, which follow the gradient of a potential that guides a robot to its goal (3). It is difficult, though, to come up with a general mechanism to escape local minima of a potential function (4) or design a potential function that has only one minimum (5). Another family of planning algorithms is composed of heuristic search techniques (e.g., A* (6)) that operate over a discretization of possible robot configurations. These algorithms provide *resolution completeness*: a path will be found if the discretization is fine enough (7, 8). A careful choice of resolution and heuristics is critical for efficient heuristic search. In principle, the classes of motion planning algorithms described above could be adapted to incorporate constraints (e.g., discretized search with constraints (9)). However, due to the complications of scaling these methods to higher-dimensional systems such as R2, they are generally not applied to modern systems.

The techniques presented so far do not consider task constraints. In general, specialized methods are required to cope with motion constraints on robotic systems. The rich history of motion constraints began in industrial control, with Cartesian constraints on manipulator end-effectors to describe assembly tasks (10–12). One of the most common constraints seen is Cartesian curve tracking, which requires a manipulator to follow a pre-planned end-effector path: a constraint on the end-effector's motion (e.g., welding and painting tasks in manufacturing). As robotic manipulators on the factory floor became more complex and had degrees of freedom redundant to the task at hand, more advanced techniques for Cartesian

curve tracking required resolution of the degrees of freedom of the robot (13) using inverse kinematic (IK) techniques (14, 15). However, using IK to generate paths is difficult, as it is hard to guarantee path continuity (16). The first applications of geometric constraints to planning techniques in low-dimensional spaces were reduced to problems of finding geodesics on polyhedral structures (17), similar to finding shortest paths of visibility graphs (18, 19). However, 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 for effective planning.

In recent years, approaches that use penalty functions and optimize over full trajectories have been proposed (20–23). These approaches relax the "hard" constraints of the task (that must be satisfied exactly, e.g., geometric task constraints, obstacle avoidance) into "soft" constraints (corresponding to some cost to optimize), combining the constraints into the formulation of the penalty function. Additionally, probabilistic completeness can still be preserved by combining optimization with random trajectory initialization in an appropriate functional space (20). However, tuning the penalty functions to avoid local minima and avoid paths that go through thin obstacles (as obstacle avoidance is relaxed) is challenging for robots with many degrees of freedom in complex scenes. This becomes even more challenging with multiple constraints and complex task goals.

Sampling-based algorithms take a very different approach. They randomly sample valid robot configurations and form a graph of valid motions (7, 8). Many algorithms provide *probabilistic completeness*: the probability of finding a solution goes to 1 with the run time of the algorithm, provided a solution exists (7, 8). 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, including very high-dimensional systems. They have also been used in very different contexts, such as computer graphics and computational structural biology (e.g., (24, 25)).

Among the classes of algorithms presented, sampling-based planning algorithms have emerged as the basis of several constrained planning approaches. This is in part because such algorithms are inherently modular and adaptive: constraints can be easily incorporated into the core of a sampling-based algorithm without affecting its method for solution finding. With these methods, systems like R2 (26) can successfully accomplish complex, constrained tasks and find motions that respect constraints. However, there have been a variety of sampling-based methods proposed to handle constraints, each with a distinct methodology for handling and incorporating constraints into the planning process.

The rest of the review is organized as follows. First, we will give a more formal description of the motion planning problem and the specification of constraints (Section 2). Next, we will provide a brief overview of the main varieties of sampling-based planning algorithms and their core components (Section 3). After that, we describe in detail the main algorithms for sampling-based planning with constraints (Section 4). This includes a description of how the core components of sampling-based planning can be modified to handle constraints. Finally, we conclude with a discussion of the the state of the art and possible avenues for future research (Section 5).

## 2. MOTION PLANNING AND CONSTRAINTS

This section develops the notation and mathematical objects necessary to understand sampling-based motion planning with constraints. Motion planning, particularly motion planning with constraints, draws from concepts in differential geometry to describe the various spaces utilized in the planning process. A good reference for these topics is (27).

This section is organized as follows. First, the mathematical notation is introduced in Section 2.1. Next, a discussion on how constraints have been expressed in the literature is given in Section 2.2. Finally, some methods for constraint composition are presented in Section 2.3.

### 2.1. Notation

The classical version of the motion planning problem can be defined as follows. One of the key ideas in motion planning is to lift the problem of planning for a dimensioned, articulated robotic system into planning for a single point that represents the robot in a higher-dimensional space. This space is called the *configuration space*, $\mathcal{Q}$: the space of all configurations for a given robot. The configuration space is a metric space (distance is defined between all points), and is usually a differentiable manifold. The dimensionality $n$ of $\mathcal{Q}$ corresponds to the number of degrees of freedom of the robot. Let $\mathcal{Q}_{\text{free}} \subseteq \mathcal{Q}$ be the *free space*: the set of configurations where the robot is not colliding with any obstacles or itself. Given a start configuration $q_{\text{start}} \in \mathcal{Q}_{\text{free}}$ and a set of goal configurations $\mathcal{Q}_{\text{goal}} \subseteq \mathcal{Q}_{\text{free}}$, the motion planning problem is then to find a continuous path $\tau : [0,1] \to \mathcal{Q}_{\text{free}}$ that connects $q_{\text{start}} = \tau(0)$ and $\tau(1) \in \mathcal{Q}_{\text{goal}}$. The algebraic complexity of $\mathcal{Q}_{\text{free}}$ determines the PSPACE-hard complexity of this problem (2). A key idea of sampling-based planning is to avoid computing $\mathcal{Q}_{\text{free}}$ exactly, described in Section 3.

**Configuration space $\mathcal{Q}$:** the space of all robot configurations

**Free space $\mathcal{Q}_{\text{free}}$:** the set of configurations where the robot is not colliding with any obstacles or itself

**Motion planning problem:** find a continuous path $\tau : [0,1] \to \mathcal{Q}_{\text{free}}$ that connects $q_{\text{start}} = \tau(0)$ and $\tau(1) \in \mathcal{Q}_{\text{goal}}$.

**Constraint function $F$:** $F : \mathcal{Q} \to \mathbb{R}^k$ such that $F(q) = \mathbf{0}$ when $q$ satisfies $k$ given constraints.

**Constraint manifold $X$:** an implicit configuration space within the ambient configuration space $\mathcal{Q}$: $X = \{\, q \in \mathcal{Q} \mid F(q) = \mathbf{0} \,\}$.

The constraints we are considering in this review are geometric, and rely only on the configuration of the robot $q$, not on other properties of the motion such as velocities or accelerations. Constraints reduce the effective number of degrees of freedom, denoted by $m$. Here, $m$ is less than $n$, the number of degrees of freedom. It is usually not possible to re-parameterize the system in terms of its effective degrees of freedom. Instead, the constraints are often written in terms of a constraint function $F : \mathcal{Q} \to \mathbb{R}^k$ such that $F(q) = \mathbf{0}$ when $q$ satisfies the constraints. Here, $k = n - m$ is the number of equality constraints imposed. Generally, $F$ must be a continuous and differentiable function. For example, take a robot with its end-effector constrained to remain at one position. That end-effector constraint could be encoded as:

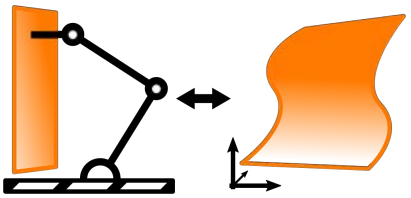
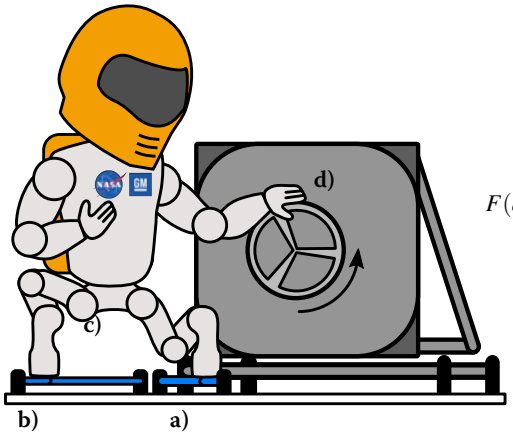


**Figure 2**

On the left, an end-effector constraint (orange plane) is imposed on a simple 3-link manipulator. On the right, the end-effector constraint corresponds to the lower-dimensional constraint manifold $X$ (orange sheet) within the manipulator's configuration space $\mathcal{Q} \subset \mathbb{R}^3$ (black axes).

$$F(q) = \text{distance of end-effector to position},$$




$$F(q) = \begin{bmatrix} \textbf{a)} \text{ distance of left foot to grasp} \\ \textbf{b)} \text{ distance of right foot to grasp} \\ \textbf{c)} \text{ angle of waist from upright} \\ \textbf{d)} \text{ distance of left hand to grasp} \end{bmatrix}$$

**Figure 3**

A breakdown of the constraints shown in Figure 1a, encoded within a constraint function. For R2 to make a full-body motion, all four constraints must be satisfied, i.e., $F(q) = \mathbf{0}$.

which evaluates to $\mathbf{0}$ when the constraint is satisfied. A more complicated example with R2 is shown in Figure 3. Inequality constraints can be dealt with in much the same way as collision-avoidance constraints, as will become clear in the next section.

The constraint function defines an $m$-dimensional implicit constrained configuration space within the ambient configuration space (shown in Figure 2):

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

which consists of all configurations that satisfy the constraint. Although hard to visualize in higher-dimensions, it is clear to see that there is a much smaller subset of allowable motion a robot can take when attempting to achieve a task, e.g., keeping a cup level or a welding tip on the surface of a piece of metal. The relative measure (volume) of $X$ compared to $Q$ is small and usually 0, which highlights the problem of using uninformed sampling to obtain valid configurations. If $F$ is continuous and differentiable and $Q$ is a differentiable manifold, then $X$ is a differentiable manifold as well. $k$, the number of equality constraints, is also referred to as the co-dimension of the manifold. The problem of motion planning with constraints can now be defined as finding a continuous path in $X_{\text{free}} = X \cap Q_{\text{free}}$ that connects a given $q_{\text{start}}$ and $q \in Q_{\text{goal}}$.

## 2.2. Constraint Expression

Specialized constraints most commonly take the form of end-effector constraints. These are constraints phrased in terms of the position and orientation of the robot's end-effector. End-effectors are generally the component of the robot that carries out the task, and as such are

usually subject to the task constraints (e.g., maintaining contact, not rotating past a limit). End-effector constraints originate in industrial control with Cartesian constraints between interacting objects (10), which was later developed into general end-effector constraints (11, 12). There are many modern incarnations of end-effector constraints (28), such as the *Task-Space Region* (TSRs) formulation (29). TSRs are a general representation for 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 loop into two manipulators that must maintain contact throughout the entire motion, closing the chain (30). Task-space regions have also been extended to *Task-Space Region Chains* (TSRCs) (29), which can model articulated kinematic structures such as doors and drawers in a scene as virtual kinematic chains. TSRCs form a closed-chain system with the robot's manipulator.

More abstractly, many approaches have been taken in the literature to specify motion constraints in the form of a constraint function $F(q)$. The most general approach is to not assume any properties of the constraint function in relation to the kinematic structure of the robot, and assume that the function encodes "distance" to the surface of the constraint manifold (31). 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. Solver speed can be improved and satisfying configurations can be generated faster if constraints can be cast as functions that can be automatically differentiated or have analytic derivatives (32).

## 2.3. Constraint Composition

The representation of a constraint function and its derivative is critical to efficiently solving an instance of a constraint motion planning problem. 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, and a visibility region that must be maintained? 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. If the constraint is phrased as a constraint function $F(q)$, multiple constraints are composed as additional equality constraints, increasing the dimension $k$ (shown in Figure 3). However, addition of new equality constraints can potentially introduce singularities in the constraint function. 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 for satisfying configurations hierarchically (33) (e.g., (26)). Another approach is to use more advanced gradient descent techniques or cyclic projection as discussed in (29). How multiple constraints are encoded has dramatic effects on the performance of a constrained motion planning method. Care must be taken when selecting an encoding, otherwise planner completeness or efficiency is sacrificed.

What if the composition of tasks we wish to achieve has more than one modality? Intermittent contact, an essential component of manipulation and legged locomotion, requires the constant addition and removal of constraints (34, 35). Combining constraints where only a subset need to be satisfied at any given time is an "or"-ing of the constraints together: creating

```
GraphPlanner(q_start, q_goal)
    G.init(q_start, q_goal);
    while no path from q_start to q_goal do
        q_rand ← Sample();
        Q ← SelectNghbrs(G, q_rand)
        for all q_near ∈ Q do
            if Connect(q_near, q_rand) then
                G.Add(q_near, q_rand)
```

```
TreePlanner(q_start, q_goal)
    T.init(q_start);
    while no path from q_start to q_goal do
        q_rand ← Sample();
        q_near ← Select(T, q_rand)
        q_new ← Extend(q_near, q_rand);
        if Connect(q_new, q_near) then
            T.Add(q_near, q_new)
```

**Figure 4**

Prototypical examples of graph- and tree-based sampling-based planners.

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 (32). 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 (36–40). Each of these planners employs domain specific knowledge to solve the problem efficiently, but no general purpose solutions have been proposed to the best of our knowledge.

## 3. SAMPLING-BASED MOTION PLANNING

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 (7, 8, 41). The general idea behind sampling-based planning is to avoid computing the free space exactly, and to instead sample free configurations and connect them to construct a tree/graph that approximates the connectivity of the underlying free space. Most sampling-based algorithms provide *probabilistic completeness* guarantees (42): if a solution exists, the probability of finding a path goes to 1 with the number of samples generated by the algorithm. If no solution exists, most sampling-based algorithms cannot recognize this (although it *is* possible in some cases (43)).

Figure 4 shows in pseudo-code the two main varieties of sampling-based planners. On the left is shown a basic version of the first sampling-based planner, the Probabilistic Roadmap Method (PRM) (44). It incrementally constructs a roadmap embedded in $\mathcal{Q}_{\text{free}}$ by repeatedly sampling collision-free configurations via rejection sampling. For each sampled configuration, it computes "nearby" configurations sampled during previous iterations. If 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 ex-

tracted via, e.g., A*. The PRM algorithm grows a roadmap that can be reused for solving many motion planning problems in the same environment.

In many cases, however, we are only interested in solving one particular problem (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 4 might be more appropriate. The most well-known variant of this type of planner is the Rapidly-exploring Random Tree (RRT) algorithm (45, 46), but several other tree-based planners have been proposed (e.g., EST (47), (48, 49)). RRT grows a tree of configurations from the start to the goal. At each iteration a random sample is generated (which may be in collision). The existing tree is extended towards the random sample from the existing configuration nearest to the sample. If the new tree branch can be connected to the goal, the algorithm terminates. A popular variant of tree planners is to grow two trees simultaneously, one from the start and one from the goal (45). Connection between the two nearest states in the two trees is tested after every tree extension. The bidirectional tree search terminates once a connection between the two trees is found.

Despite their differences, many sampling-based planners have similar requirements from the robot's configuration space. Below are some of the components that are commonly used in sampling-based algorithms. Note that this list is not a complete listing of all sampling-based planner components, but a listing of important components for constrained sampling-based planning.

**Samplers** 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 (50, 51). Alternatively, one could sample near the medial axis, a one-dimensional structure formed by all configurations that have more than one closest point on the boundary of $Q_{\text{free}}$, i.e., exactly between two obstacles. Configurations on the medial axis tend to "see" more of $Q_{\text{free}}$ than other configurations (52, 53). There also has been work in sampling entire lower-dimensional manifolds of the configuration space (54), which can better capture the connectivity of the free space in some problems. Finally, deterministic quasi-random samples have been shown to improve the spread of samples (dispersion) compared to uniformly random sampling (55).

**Metrics & nearest-neighbor data structures** 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 (56).

**Local planner** A local planner is a fast, not necessarily complete method for finding paths between nearby configurations. In many cases interpolation is used (or SLERP for rota-

tions (57)). Tree-based planners can also be easily extended for kinodynamic motion planning, where the dynamics can often be written as $\dot{q} = f(q,u)$. During the extension, a steering function that drives the system towards a randomly sampled state is used as a local planner. Randomly sampling a control input $u$ is generally sufficient for probabilistic completeness (46).

**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 (49, 58). Although random projections often work well in practice (59), for constrained planning it may be difficult to define a projection over $Q$ that approximates the density of sampling in the implicit configuration space $X$.

The Open Motion Planning Library (60) provides various implementations of these core components as well as implementations of all the sampling-based planning algorithms cited in this section.

The paths produced by sampling-based planning are feasible, but sometimes far from optimal. There are various techniques that post-process paths to optimize them locally (61). 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 (62). Subsequent work has improved the convergence rate (see, e.g., (63)), 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 (64, 65). Asymptotically optimality can be extended to kinodynamic planning planning (66–68). It is also possible to create sparse roadmaps or trees that guarantee asymptotic *near*-optimality (68, 69). 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 are highly problem-dependent. Systematic benchmarking is needed to determine a good algorithm for a given problem domain (70).

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 (58, 71). 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.

## 4. SAMPLING-BASED MOTION PLANNING WITH CONSTRAINTS

### 4.1. Challenges

Constraints introduce anther element of difficulty to the problem: the need to find configurations that satisfy the constraint function. The core concepts that enable a sampling-based planner to perform effectively 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. The structure of the implicit region defined by a constraint function is not known *a priori*, and is thus hard to sample from without careful consideration or pre-processing. A planner needs to be able to either sample configurations that satisfy the constraint function (solutions to $F(q) = \mathbf{0}$) or guide the search towards valid regions of the space.

**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 the "swiss roll" function (72). A more appropriate metric for this space would be something like the Riemannian metric, as the constraint function usually defines a Riemannian manifold (27). However, implicitly defined metrics such as the Riemannian metric are expensive to compute as they require 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 we already had a roadmap 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 (72). 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 (73). Unfortunately, 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*. On manifolds, there are many existing approaches within the literature to compute the minimum length geodesic (74–77). 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 authors' 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, sampling and local planning must be augmented to satisfy constraints, as these both directly affect whether the planning generates a valid path. As such, most methods focus on these two elements. In existing work, the metric from the ambient configuration space is typically used, which works well in practi-

cal applications. The ambient configuration space's metric defines a semi-metric (78) for the constrained space, as the triangle inequality may not hold given sufficient curvature of the implicit space. However, semi-metrics are "good enough" for most sampling-based planners as this discrepancy only affects the effectiveness of exploration. In this case, some theoretical guarantees may not hold.

## 4.2. Methodology 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 families of methods lie upon the spectrum 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. 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 and 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. 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 4.3. 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

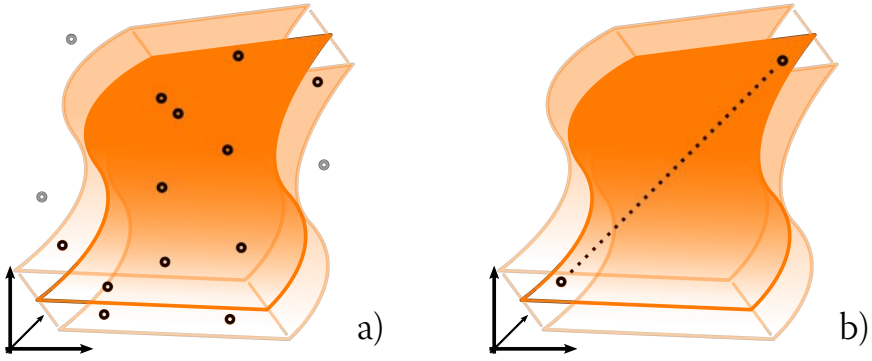



**Figure 5**

Sampling and local planning with relaxation-based constraint handling. The constraint manifold (orange) 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).

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., (47–49)) might have an advantage, assuming they can be adapted to deal with constraints. Some recent work (79) described in Section 4.3.5 suggests that this is the case for some systems, but further study is needed.

## 4.3. Methodologies

**4.3.1. Relaxation.** 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 uninformed 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, $\|F(q)\| < \varepsilon$. 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 5a. This technique is applied by (80, 81) for bi-manual manipula-

tion problems. These works leverage execution-level controllers with compliant, closed-chain control to ensure successful execution despite using 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 (82, 83). Local planning is also very simple within a relaxation-based method, shown in Figure 5b. 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.

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 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 suit the planning problem could be used to solve relaxed constrained planning problems. Sampling-based planners also retain their probabilistic completeness when using the relaxed constraint. However, execution success is no longer a given, as execution success is now determined by the controller's 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.

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 more than other techniques.

**4.3.2. Projection.** 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 configurations is with a *projection operator*. 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 $P(q) : \mathcal{Q} \to X$, where if $q \in X, P(q) = q$. 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 $J(q)$ (15, 84). This is shown in Algorithm 1.

### Projection Through Iteration

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** $q$

**Algorithm 1.** 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 $\Delta q$ to $J(q)\Delta q = F(q)$ can be found. QR factorization (85) and other matrix decompositions might be more efficient with equivalent performance for certain problems.

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 6a. 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, as proved in (29). This property of projection sampling preserves the probabilistic completeness of RRT-like projection-based algorithms (29). An example of local planning using a projection operator is shown in Figure 6b. 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 (30), which considers constrained motion planning for closed-chain planar chains. Many modern methods (e.g., (29, 32)) adopted this approach for local planning. Recently, work has gone into investigating continuous local planning using projection (77, 86), avoiding issues of discontinuity found with naïve discretization of the geodesic.

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 (87), and complex loop-closure problems in robotics were solved with PRM variants using active/passive chain methods (30, 88). 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. Projection operators were also used to solve curve tracking problems in early industrial applications (89).

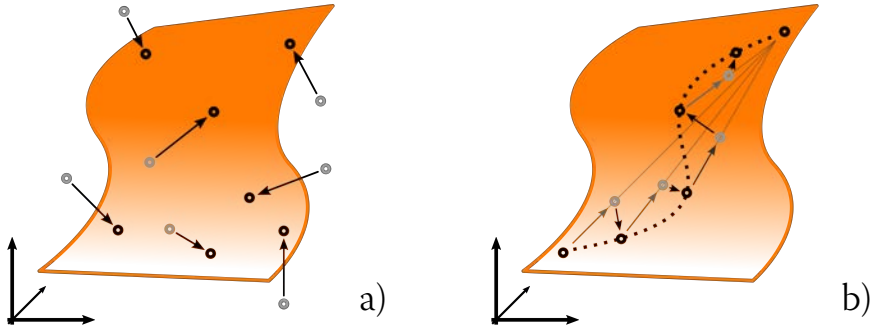



**Figure 6**

Sampling and local planning with projection-based constraint handling. The constraint manifold (orange) 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 (grey 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.

The idea of projection to satisfy constraints was applied to general end-effector constraints in (90). Task Constrained RRT (28) further generalized the idea of constraints and utilized Jacobian gradient descent (15) for projection. Recently, CBIRRT2 (29), the motion planner implemented for the Humanoid Path Planner System (32), and other planners such as one for the HRP2 humanoid (91) utilize projection with general constraints.

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 homotopy 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 (34). In a foliation, a manifold $X$ is decomposed into a disjoint union of connected submanifolds called *leaves*. 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. Its tangent space is the nullspace of the manipulator Jacobian. This property has been exploited for manipulation planning (34) and by a few constrained planners (92, 93) 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.

**Self-motion manifold:** The set of configurations that all result in the same end-effector pose for a redundant manipulator. Its tangent space is the nullspace of the manipulator Jacobian.

Projection-based planners have a number of notable advantages that contribute to their success as one of the most widely implemented methodologies. Primarily, the projection op-

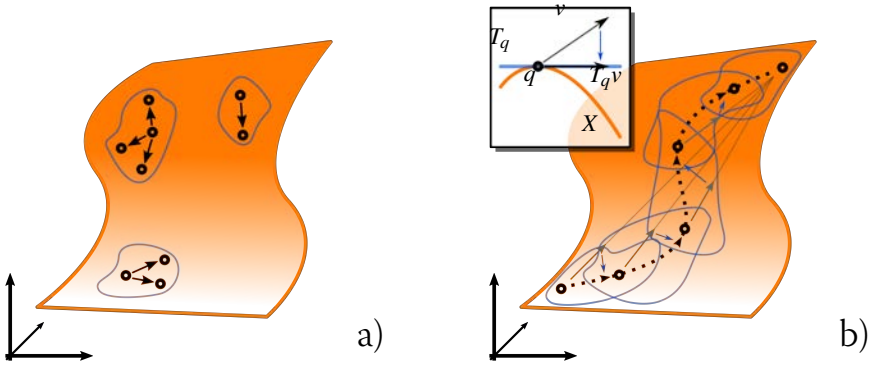



**Figure 7**

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 (blue 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 $v$ (grey arrow) and projected (blue 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.

erator 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 (30), 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 (26, 29). 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.

**4.3.3. Tangent Spaces.** 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. 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 $q \in X$, with an $n \times (n-k)$ orthonormal basis $\Phi_q$. 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 techniques 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 7a. 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 7b. From an initial configuration, a vector to the goal configuration is computed. This vector is projected into the tangent space and decomposed 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.

**Tangent space:** A real vector space which contains all possible *tangent* motions to a point upon the constraint manifold.

Projection from tangent spaces was utilized within the work of (30) to generate nearby samples, which are then fixed up with projection. Tangent spaces have been used by (28, 94) for manipulators under general end-effector constraints. The technique has also seen many applications in curve tracking constraints for redundant manipulators (95–97) and structural biology to generate valid motions of proteins with loop closures (98–100).

In tangent space-based techniques, 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 typically can be projected into the satisfying set with 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 (28, 94).

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.

**4.3.4. Atlas.** Furthering the idea of 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 (27). Such methods also require that the constraint function defines a manifold. Unlike methods described in Section 4.3.3, atlas-based methods store generated tangent spaces to avoid re-computation. 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 (101). The key difference between the method described in (101) and atlas-based planners is the incremental

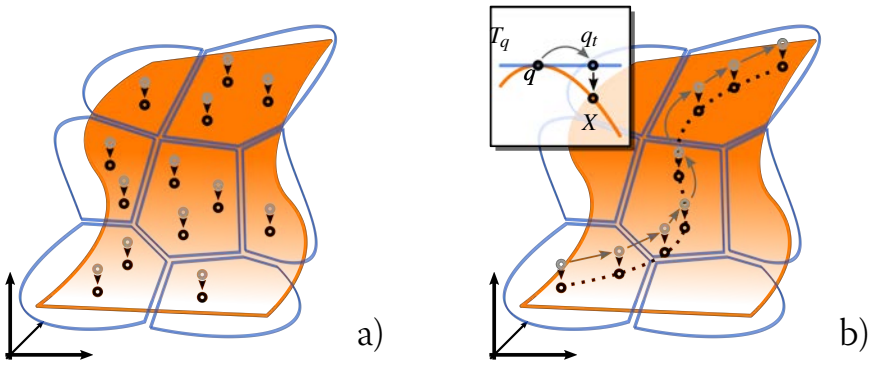



**Figure 8**

Atlas-based sampling and local planning, akin to AtlasRRT. The atlas is a set of tangent polytopes covers the constraint manifold (blue 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), $\psi_q$. Local planning is shown in **b)**. Roughly, interpolation is done with the tangent space (grey arrows) from a configuration $q$ to another $q_t$, and new configurations are projected to the manifold for validation (black arrows). This continues until the goal is reached. See (31) for details.

construction of the atlas interleaved with space exploration. These tangent spaces are generated and utilized exactly as described above in Section 4.3.3, and allow sampling and mapping to and from the manifold and the tangent space. 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. A point $t \in T_q$ can be mapped into $q_t \in Q$ by $q_t = q + \Phi_q t$. To map the configuration $q_t$ onto the manifold (an exponential map $\psi_q$), 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 (102), along with other operations on implicit manifolds.

**Atlas:** A concept borrowed from differential geometry (27) where a manifold is modeled as a collection of *charts*, each of which can be approximated by a tangent space.

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 4.3.3. This is shown in Figure 8a. Planners that implement variants of the atlas-based methodology are derived from the procedure described in (101). AtlasRRT (31) 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 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 8b. Tangent Bundle RRT, or TB-RRT (103) 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 (104) 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* (105) with AtlasRRT* (106). 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 (107). Additionally, the methodology behind building an atlas incrementally has been extended to general sampling-based motion planning in (108), which enables any sampling-based planner to build an atlas approximation while planning with constraints. The extensions made to AtlasRRT underline the importance of using sampling-based methods for planning with constraints, as the methods are modular, easily adapted to new problem instances, and extended with features such as asymptotic optimality.

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 ring 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. Atlas-based approaches are also probabilistically complete (31).

The primary downside to atlas-based approaches is the difficulty of implementation, as the atlas data-structure needs to be efficient and correct. 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 (108). Maintaining an approximation of the manifold is computationally inefficient for constraints that only have a few equality constraints relative to

the configuration space. The tangent 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.

**4.3.5. Implicit Space Representation.** The techniques discussed above cover part of the spectrum of methods to compute satisfying configurations for constrained motion planning. 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. Instead, these methods augment primitive operations (e.g., samplers and local planners, as discussed) 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. 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 planning problem requires design of a bespoke planner that integrates the two. A recent work (79) proposes a general framework for constrained sampling-based planning, which approaches augmenting primitive operations not from within the planner, but from within the representation of the configuration space, leveraging the modularity of sampling-based planners. This has the benefit of enabling composition of any emulated constraint methodology with a broad class of sampling-based planners, allowing choice of the combination best suited to a problem at the cost of losing potential benefits that coupled implementation can bring (e.g., speed, leveraging planner properties). The benefits of this unified approach can be considerable in some cases. Figure 9 shows a system described in more detail in (79). Here, the combination of the KPIECE planner (49) planner and the projection-based constraint methodology was shown to be orders of magnitude faster than using RRT-Connect (45), which was the planner modified in prior works to accomodate constraints (29, 31, 103).

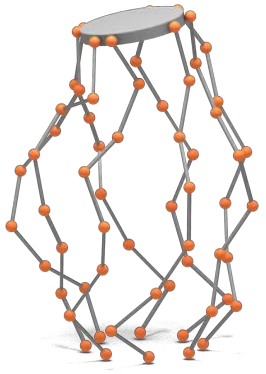


**Figure 9**

A parallel manipulator with 168 degrees of freedom. The unified approach allows use of almost any sampling-based planner for this high-dimensional system (79).

**4.3.6. Reparameterization.** In some cases, the constraint function and manipulator topology lend themselves to *reparameterization*, where another configuration space is generated for the robot and constraint. Configurations within this newly generated space fully describe the robot's state and satisfy the constraint. This allows for any traditional planner to be utilized on top of a new configuration space, enabling the machinery of the planner to function unaf-

fected while satisfying constraints. Tangent space- and atlas-based approaches can be thought of as reparameterization-based approaches, but a distinction made with our organization as presented here is the idea of pre-computation versus online exploration and constructions. The reparameterization-based approaches presented in this subsection are usually computed before planning, while tangent space-based approaches are generally generated online for computational efficiency. Additionally, reparameterization is distinct from the tangent space and atlas approaches. Reparameterization creates a global, non-linearly-related space while tangent space- and atlas-based methods create local, linear approximations.

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-chain systems (109) and reachable volume space (110) for general kinematic chains. Generally, these methods have their own methods for sampling within their reparameterized space (shown in Figure 10a), and their own methods of stepping within the reparameterized space (shown in Figure 10b). Deformation space reparameterizes closed-chain planar systems by encoding the "deformation" of the closed-chain 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 (7), 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) (110).

Reparameterization-based approaches are appealing from a sampling-based planning perspective. If it was possible to simply plan within a 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 pre-computation 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 problem domain (e.g., planar chains, closed loop systems) and are generally complex to implement, which prevents wide-spread applicability to many different robotic systems.

**4.3.7. 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 (29). First, samples are drawn to cover the area of interest in the satisfying sub-

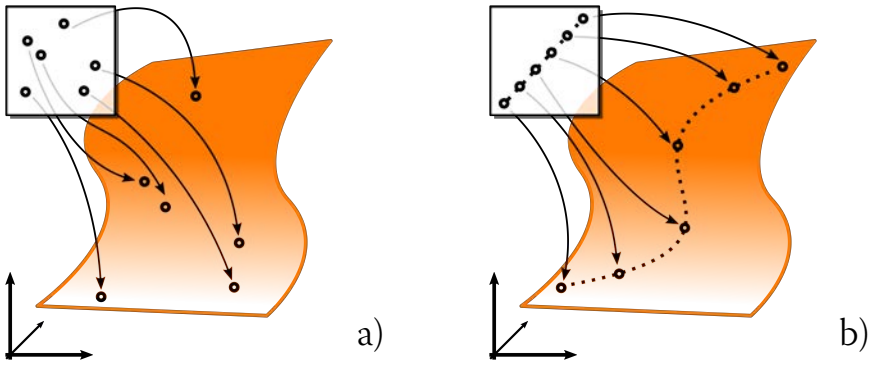



**Figure 10**

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.

set 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 (111, 112) 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 (113, 114), 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 pre-computation, 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, pre-computation might be the correct answer to avoid repeating computation online. Additionally, pre-computation-based approaches that apply to changing environments require an element of online planning to handle changing obstacle configurations and potentially invalidated edges in an offline-computed roadmap.

## 5. DISCUSSION

This survey has covered the large variety of methods that have been proposed to allow sampling-based planning algorithms to incorporate geometric motion constraints. These methods have been shown to be effective on many real-world scenarios. However, as of yet there is 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 in this review, uniform sampling on implicit spaces 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. Additionally, future works should further investigate primitive operations that better represent the underlying constraint manifold, such as using distance metrics that capture the curvature of the manifold, local planners that generate continuous paths, and samplers that can approach uniform sampling of the manifold.

Another avenue for future work is addressing forces while planning with constraints. Intrinsic to many 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 constraint. For the approaches presented in this work, the constraint is translated into a geometric constraint, because in general kinodynamic planning (i.e., with forces and dynamics) is much more complicated than planning quasi-statically. A constrained sampling-based approach that leverages information from its constraint methodology could be potentially helpful and make force computations feasible. Finally, there is interesting future research into 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 would require new techniques to simultaneously explore different constraint manifolds as well as ways to transition between them.

### SUMMARY POINTS

1. Several different methods have been proposed to extend sampling-based algorithms to incorporate geometric constraints on robot motion.
2. These methods are focused primarily on sampling and interpolation on constraint manifolds.
3. Five categories of constraint methodologies were identified: 1) relaxation, 2) projection, 3) tangent space sampling, 4) incremental atlas construction, and 5) reparameterization.
4. Unification of methodologies 2–4 is possible by using a representation of the implicit space.

### FUTURE ISSUES

1. Further study is needed to evaluate the relative merits of each of the constraint methodologies for sampling-based motion planning algorithms.
2. A general method is needed to handle multi-modal constrained planning (such as locomotion and manipulation planning), where a planning algorithm needs to explore several different constraint manifolds and possible transitions between them.
3. Application of force is intrinsic to constraints: future methods should leverage information about constraints to consider the forces that will be applied by a robot while planning.
4. More work needs to be done creating primitive operations that better represent constraint spaces (e.g., local planners that are continuous, metrics for implicit spaces, manifold sampling).

## DISCLOSURE STATEMENT

The authors are not aware of any affiliations, memberships, funding, or financial holdings that might be perceived as affecting the objectivity of this review.

## ACKNOWLEDGMENTS

Work on this paper by Zachary Kingston, Mark Moll and Lydia E. Kavraki has been supported in part by NSF IIS 1317849, NSF IIS 1718478, and Rice University funds. Zachary Kingston is also supported by a NASA Space Technology Research Fellowship 80NSSC17K0163.

## LITERATURE CITED

1. Diftler MA, Mehling JS, Abdallah ME, Radford NA, Bridgwater LB, et al. 2011. Robonaut 2 — the first humanoid robot in space. In *IEEE Intl. Conf. on Robotics and Automation*. 2178–2183 pp.
2. Canny JF. 1988. *The Complexity of Robot Motion Planning*. MIT Press
3. Khatib O. 1986. Real-time obstacle avoidance for manipulators and mobile robots. *Intl. J. of Robotics Research* 5:90–98
4. Barraquand J, Latombe JC. 1991. Robot motion planning : A distributed representation approach. *Intl. J. of Robotics Research* 10:628–649
5. Rimon E, Koditschek DE. 1992. Exact robot navigation using artificial potential functions. *IEEE Trans. on Robotics and Automation* 8:501–518
6. Aine S, Swaminathan S, Narayanan V, Hwang V, Likhachev M. 2015. Multi-heuristic a*. *Intl. J. of Robotics Research*
7. Choset H, Lynch KM, Hutchinson S, Kantor G, Burgard W, et al. 2005. *Principles of Robot Motion: Theory, Algorithms, and Implementations*. MIT Press
8. LaValle SM. 2006. *Planning Algorithms*. Cambridge University Press

9. Phillips M, Hwang V, Chitta S, Likhachev M. 2016. Learning to plan for constrained manipulation from demonstrations. *Autonomous Robots* 40:109–124
10. Ambler AP, Popplestone RJ. 1975. Inferring the positions of bodies from specified spatial relationships. *Artificial Intelligence* 6:157–174
11. Mason MT. 1981. Compliance and force control for computer controlled manipulators. *IEEE Transactions on Systems, Man, and Cybernetics* 11:418–432
12. Khatib O. 1987. A unified approach for motion and force control of robot manipulators: The operational space formulation. *IEEE Journal Robotics Automation* 3:43–53
13. Seereeram S, Wen JT. 1995. A global approach to path planning for redundant manipulators. *IEEE Transactions on Robotics and Automation* 11:152–160
14. Whitney DE. 1969. Resolved motion rate control of manipulators and human prostheses. *IEEE Transactions on Man-Machine Systems* 10:47–53
15. Buss SR, Kim JS. 2005. Selectively damped least squares for inverse kinematics. *Journal of Graphics, GPU, and Game Tools* 10:37–49
16. Beeson P, Hart S, Gee S. 2016. Cartesian motion planning & task programming with CRAFTSMAN. In *Robotics: Science and Systems Workshop on Task and Motion Planning*
17. Mitchell JSB, Mount DM, Papdimitrious CH. 1987. The discrete geodesic problem. *SIAM Journal on Computing* 16:647–668
18. Asano T, Asano T, Guibas L, Hershberger J, Imai H. 1985. Visibility-polygon search and euclidean shortest paths. In *Annual Symp. on Foundations of Computer Science*. 155–164 pp.
19. Alexopoulos C, Griffin PM. 1992. Path planning for a mobile robot. *IEEE Transactions on Systems, Man, and Cybernetics* 22:318–322
20. Zucker M, Ratliff N, Dragan A, et al. 2013. CHOMP: Covariant hamiltonian optimization for motion planning. *International Journal of Robotics Research* 32:1164–1193
21. Kalakrishnan M, Chitta S, Theodorou E, Pastor P, Schaal S. 2011. STOMP: Stochastic trajectory optimization for motion planning. In *IEEE International Conference on Robotics and Automation*. 4569–4574 pp.
22. Schulman J, Duan Y, Ho J, et al. 2014. Motion planning with sequential convex optimization and convex collision checking. *International Journal of Robotics Research* 33:1251–1270
23. Dong J, Mukadam M, Dellaert F, Boots B. 2016. Motion planning as probabilistic inference using Gaussian processes and factor graphs. In *Robotics: Science and Systems*
24. Latombe JC. 1999. Motion planning: A journey of robots, molecules, digital actors, and other artifacts. *International Journal of Robotics Research* 18:1119–1128
25. Gipson B, Hsu D, Kavraki LE, Latombe JC. 2012. Computational models of protein kinematics and dynamics: Beyond simulation. *Annual Review of Analytical Chemistry* 5:273–291
26. Baker W, Kingston Z, Moll M, Badger J, Kavraki LE. 2017. Robonaut 2 and you: Specifying and executing complex operations. In *IEEE Workshop on Advanced Robotics and its Social Impacts*
27. Spivak M. 1999. *A Comprehensive Introduction to Differential Geometry*. Publish or Perish
28. Stilman M. 2010. Global manipulation planning in robot joint space with task constraints. *IEEE Transactions on Robotics and* 26:576–584
29. Berenson D. 2011. Constrained manipulation planning. Ph.D. thesis, Carnegie Mellon University, Pittsburgh, PA
30. Yakey JH, LaValle SM, Kavraki LE. 2001. Randomized path planning for linkages with closed kinematic chains. *IEEE Transactions on Robotics and Automation* 17:951–958
31. Jaillet L, Porta JM. 2013. Path planning under kinematic constraints by rapidly exploring manifolds. *IEEE Transactions on Robotics and* 29:105–117

32. Mirabel J, Tonneau S, Fernbach P, et al. 2016. HPP: A new software for constrained motion planning. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*. 383–389 pp.
33. Sentis L, Khatib O. 2005. Synthesis of whole-body behaviors through hierarchical control of behavioral primitives. *International Journal of Humanoid Robotics* 2:505–518
34. Siméon T, Laumond JC, Cortés J, Sahbani A. 2004. Manipulation planning with probabilistic roadmaps. *International Journal of Robotics Research* 32:729–746
35. Hauser K, Bretl T, Latombe JC, Harada K, Wilcox B. November/December 2008. Motion planning for legged robots on varied terrain. *Intl. J. of Robotics Research* 27:1325–1349
36. Perrin N, Stasse O, Lamiraux F, Kim YJ, Manocha D. 2012. Real-time footstep planning for humanoid robots among 3d obstacles using a hybrid bounding box. In *IEEE Intl. Conf. on Robotics and Automation*. 977–982 pp.
37. Reid W, Fitch R, Göktogan AH, Sukkarieh S. 2016. Motion planning for reconfigurable mobile robots using hierarchical fast marching trees. In *Workshop on the Algorithmic Foundations of Robotics*
38. Dantam NT, Kingston ZK, Chaudhuri S, Kavraki LE. 2016. Incremental task and motion planning: A constraint-based approach. In *Robotics: Science and Systems*
39. Lozano-Pérez T, Kaelbling LP. 2014. A constraint-based method for solving sequential manipulation planning problems. In *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems*. 3684–3691 pp.
40. Kaelbling L, Lozano-Perez T. 2011. Hierarchical task and motion planning in the now. In *IEEE Intl. Conf. on Robotics and Automation*. 1470–1477 pp.
41. Elbanhawi M, Simic M. 2014. Sampling-based robot motion planning: A review. *IEEE Access* 2:56–77
42. Ladd AM, Kavraki LE. 2004. Measure theoretic analysis of probabilistic path planning. *IEEE Trans. on Robotics and Automation* 20:229–242
43. McCarthy Z, Bretl T, Hutchinson S. 2012. Proving path non-existence using sampling and alpha shapes. In *IEEE Intl. Conf. on Robotics and Automation*. 2563–2569 pp.
44. Kavraki LE, Švestka P, Latombe JC, Overmars M. 1996. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE Transactions on Robotics and Automation* 12:566–580
45. Kuffner J, LaValle SM. 2000. RRT-Connect: An efficient approach to single-query path planning. In *IEEE Intl. Conf. on Robotics and Automation*. San Francisco, CA, 995–1001 pp.
46. LaValle SM, Kuffner JJ. 2001. Randomized kinodynamic planning. *International Journal of Robotics Research* 20:378–400
47. Hsu D, Latombe JC, Motwani R. 1999. Path planning in expansive configuration spaces. *Intl. J. of Computational Geometry and Applications* 9:495–512
48. Ladd AM, Kavraki LE. 2005. Motion planning in the presence of drift, underactuation and discrete system changes. In *Robotics: Science and Systems I*. Boston, MA: MIT Press, 233–241 pp.
49. Şucan IA, Kavraki LE. 2012. A sampling-based tree planner for systems with complex dynamics. *IEEE Trans. on Robotics* 28:116–131
50. Amato N, Bayazit O, Dale L, Jones C, Vallejo D. 1999. OBPRM: An obstacle-based PRM for 3D workspaces. In *Robotics: The Algorithmic Perspective*, ed. PK Agarwal, LE Kavraki, MT Mason, pp. 155–168. A.K. Peters
51. Boor V, Overmars MH, van der Stappen AF. 1999. The Gaussian sampling strategy for probabilistic roadmap planners. In *Proc. 1999 IEEE Intl. Conf. on Robotics and Automation*. 1018–1023 pp.
52. Wilmarth SA, Amato N, Stiller PF. 1999. MAPRM: A probabilistic roadmap planner with sam-

pling on the medial axis of the free space. In *Proc. 1999 IEEE Intl. Conf. on Robotics and Automation*. Detroit, MI, 1024–1031 pp.
53. Hsu D, Jiang T, Reif J, Sun Z. 2003. The bridge test for sampling narrow passages with probabilistic roadmap planners. *Proc. 2003 IEEE Intl. Conf. on Robotics and Automation* 3:4420–4426
54. Salzman O, Hemmer M, Halperin D. 2015. On the power of manifold samples in exploring configuration spaces and the dimensionality of narrow passages. *IEEE Transactions on Automation Science and Engineering* 12:529–538
55. LaValle SM, Branicky MS, Lindemann SR. 2004. On the relationship between classical grid search and probabilistic roadmaps. *Intl. J. of Robotics Research* 23:673
56. Andoni A, Indyk P. 2017. Nearest neighbours in high-dimensional spaces. In *Handbook of Discrete and Computational Geometry*, ed. JE Goodman, J O'Rourke, CD Tóth. CRC Press, 3rd ed.
57. Shoemake K. 1985. Animating rotation with quaternion curves. In *ACM SIGGRAPH Computer Graphics*, vol. 19. ACM, 245–254 pp.
58. Sánchez G, Latombe JC. 2001. A single-query bi-directional probabilistic roadmap planner with lazy collision checking. In *The Tenth International Symposium on Robotics Research*. 403–417 pp.
59. Şucan IA, Kavraki LE. 2009. On the performance of random linear projections for sampling-based motion planning. In *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems*. 2434–2439 pp.
60. Şucan IA, Moll M, Kavraki LE. 2012. The Open Motion Planning Library. *IEEE Robotics & Automation Magazine* 19:72–82. http://ompl.kavrakilab.org
61. Geraerts R, Overmars M. 2007. Creating high-quality paths for motion planning. *Intl. J. of Robotics Research* 26:845–863
62. Karaman S, Frazzoli E. 2011. Sampling-based algorithms for optimal motion planning. *Intl. J. of Robotics Research* 30:846–894
63. Gammell JD, Srinivasa SS, Barfoot TD. 2015. Batch informed trees (BIT*): Sampling-based optimal planning via the heuristically guided search of implicit random geometric graphs. In *IEEE Intl. Conf. on Robotics and Automation*. 3067–3074 pp.
64. Luna R, Şucan IA, Moll M, Kavraki LE. 2013. Anytime solution optimization for sampling-based motion planning. In *IEEE Intl. Conf. on Robotics and Automation*. 5053–5059 pp.
65. Luo J, Hauser K. 2014. An empirical study of optimal motion planning. In *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems*
66. Webb DJ, van den Berg J. 2013. Kinodynamic RRT*: Asymptotically optimal motion planning for robots with linear dynamics. In *IEEE Intl. Conf. on Robotics and Automation*. 5054–5061 pp.
67. Hauser K, Zhou Y. 2016. Asymptotically optimal planning by feasible kinodynamic planning in a state–cost space. *IEEE Trans. on Robotics* 32:1431–1443
68. Li Y, Littlefield Z, Bekris KE. 2016. Asymptotically optimal sampling-based kinodynamic planning. *International Journal of Robotics Research* 35:528–564
69. Dobson A, Bekris KE. 2014. Sparse roadmap spanners for asymptotically near-optimal motion planning. *International Journal of Robotics Research* 33:18–47
70. Moll M, Şucan IA, Kavraki LE. 2015. Benchmarking motion planning algorithms: An extensible infrastructure for analysis and visualization. *IEEE Robotics & Automation Magazine (Special Issue on Replicable and Measurable Robotics Research)* 22:96–102
71. Bohlin R, Kavraki LE. 2000. Path planning using lazy PRM. In *Proc. 2000 IEEE Intl. Conf. on Robotics and Automation*. San Francisco, CA, 521–528 pp.
72. Tenenbaum JB, de Silva V, Langford JC. 2000. A global geometric framework for nonlinear dimensionality reduction. *Science* 290:2319–2323

73. Chaudhry R, Ivanov Y. 2010. Fast approximate nearest neighbor methods for non-Euclidean manifolds with applications to human activity analysis in videos. In *European Conference on Comput. Vision*. Springer, 735–748 pp.
74. Kimmel R, Sethian JA. 1998. Computing geodesic paths on manifolds. *Proc. National Academy of Sciences* 95:8431–8435
75. Ying L, Candes EJ. 2006. Fast geodesic computation with the phase flow method. *Journal of Computational Physics* 220:6–18
76. Crane K, Weischedel C, Wardetzky M. 2013. Geodesics in heat: A new approach to computing distance based on heat flow. *ACM Transactions on Graphics* 32:152:1–152:11
77. Hauser K. 2014. Fast interpolation and time-optimization with contact. *The International Journal of Robotics Research* 33:1231–1250
78. Deza MM, Deza E. 2016. *Encyclopedia of Distances*. Springer
79. Kingston Z, Moll M, Kavraki LE. 2017. Decoupling constraints from sampling-based planners. In *International Symposium of Robotics Research*
80. Bonilla M, Farnioli E, Pallottino L, Bicchi A. 2015. Sample-based motion planning for soft robot manipulators under task constraints. In *IEEE International Conference on Robotics and Automation*. 2522–2527 pp.
81. Bonilla M, Pallottino L, Bicchi A. 2017. Noninteracting constrained motion planning and control for robot manipulators. In *IEEE Intl. Conf. on Robotics and Automation*. 4038–4043 pp.
82. Rodriguez S, Thomas S, Pearce R, Amato NM. 2008. RESAMPL: A region-sensitive adaptive motion planner. *Algorithmic Foundation of Robotics VII* :285–300
83. Bialkowski J, Otte M, Frazzoli E. 2013. Free-configuration biased sampling for motion planning. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*. 1272–1279 pp.
84. Dennis J, Schnabel R. 1983. *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*. Englewood Cliffs, NJ: Prentice-Hall
85. Press WH, Teukolsky SA, Vetterling WT, Flannery BP. 1992. *Numerical Recipes in C: The art of Scientific Computing*. Cambridge University Press, 2nd ed.
86. Mirabel J, Lamiraux F. 2017. Manipulation planning: building paths on constrained manifolds. Tech. rep., LAAS-GEPETTO - Équipe Mouvement des Systèmes Anthropomorphes
87. Wedemeyer WJ, Scheraga HA. 1999. Exact analytical loop closure in proteins using polynomial equations. *Journal of Computational Chemistry* 20:819–844
88. Cortés J, Siméon T, Laumond JP. 2002. A random loop generator for planning the motions of closed kinematic chains using PRM methods. In *IEEE International Conference on Robotics and Automation*. 2141–2146 pp.
89. Oriolo G, Ottavi M, Vendittelli M. 2002. Probabilistic motion planning for redundant robots along given end-effector paths. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*. 1657–1662 pp.
90. Yao Z, Gupta K. 2005. Path planning with general end-effector constraints: Using task space to guide configuration space search. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*. 1875–1880 pp.
91. Kanehiro F, Yoshida E, Yokoi K. 2012. Efficient reaching motion planning and execution for exploration by humanoid robots. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*. 1911–1916 pp.
92. Kim J, Ko I, Park FC. 2016. Randomized path planning for tasks requiring the release and regrasp of objects. *Advanced Robotics* 30:270–283
93. Xian Z, Lertkultanon P, Pham QC. 2017. Closed-chain manipulation of large objects by multi-arm

robotic systems. *IEEE Robotics and Automation Letters* 2:1832–1839
94. Weghe MV, Ferguson D, Srinivasa SS. 2007. Randomized path planning for redundant manipulators without inverse kinematics. In *IEEE-RAS International Conference on Humanoid Robots*. 477–482 pp.
95. Oriolo G, Vendittelli M. 2009. A control-based approach to task-constrained motion planning. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*. 297–302 pp.
96. Vendittelli M, Oriolo G. 2009. Task-constrained motion planning for underactuated robots. In *IEEE International Conference on Robotics and Automation*. 2965–2970 pp.
97. Cefalo M, Oriolo G, Vendittelli M. 2013. Task-constrained motion planning with moving obstacles. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*. 5758–5763 pp.
98. Zhang Y, Hauser K. 2013. Unbiased, scalable sampling of protein loop conformations from probabilistic priors. *BMC Structural Biology* 13:S9
99. Pachov DV, van den Bedem H. 2015. Nullspace sampling with holonomic constraints reveals molecular mechanisms of protein G$\alpha$s. *PLoS Computational Biology* 11:e1004361
100. Fonseca R, Budday D, van dem Bedem H. 2016. Collision-free poisson motion planning in ultra high-dimensional molecular conformation spaces. *arXiv preprint*
101. Henderson ME. 2002. Multiple parameter continuation: Computing implicitly defined *k*-manifolds. *International Journal of Bifurcation and Chaos* 12:451–476
102. Rheinboldt WC. 1996. MANPAK: A set of algorithms for computations on implicitly defined manifolds. *Computers & Mathematics with Applications* 32:15–28
103. Kim B, Um TT, Suh C, Park FC. 2016. Tangent bundle RRT: A randomized algorithm for constrained motion planning. *Robotica* 34:202–225
104. Bohigas O, Henderson ME, Ros L, Manubens M, Porta JM. 2013. Planning singularity-free paths on closed-chain manipulators. *IEEE Transactions on Robotics and* 29:888–898
105. Karaman S, Frazzoli E. 2011. Sampling-based algorithms for optimal motion planning. *International Journal of Robotics Research* 30:846–894
106. Jaillet L, Porta JM. 2013. Asymptotically-optimal path planning on manifolds. *Robotics: Science and Systems*
107. Bordalba R, Ros L, Porta JM. 2017. Kinodynamic planning on constraint manifolds. *ArXiv e-prints*
108. Voss C, Moll M, Kavraki LE. 2017. Atlas + X: Sampling-based planners on constraint manifolds. Tech. Rep. 17-02, Department of Computer Science, Rice University, Houston, TX
109. Han L, Rudolph L, Blumenthal J, Valodzin I. 2008. Convexly stratified deformation spaces and efficient path planning for planar closed chains with revolute joints. *International Journal of Robotics Research* 27:1189–1212
110. McMahon T. 2016. Sampling based motion planning with reachable volumes. Ph.D. thesis, Texas A&M University
111. Kuffner JJ, Kagami S, Nishiwaki K, Inaba M, Inoue H. 2002. Dynamically-stable motion planning for humanoid robots. *Autonomous Robots* 12:105–118
112. Burget F, Hornung A, Bennewitz M. 2013. Whole-body motion planning for manipulation of articulated objects. In *IEEE International Conference on Robotics and Automation*. 1656–1662 pp.
113. Igarashi T, Stilman M. 2010. Homotopic path planning on manifolds for cabled mobile robots. In *International Workshop on the Algorithmic Foundations of Robotics*. 1–18 pp.
114. Şucan IA, Chitta S. 2012. Motion planning with constraints using configuration space approximations. *IEEE/RSJ International Conference on Intelligent Robots and Systems* :1904–1910
115. Coleman D, Şucan IA, Moll M, Okada K, Correll N. 2015. Experience-based planning with sparse

roadmap spanners. In *IEEE Intl. Conf. on Robotics and Automation*. 900–905 pp.
116. Hauser K. 2016. Continuous pseudoinversion of a multivariate function: Application to global redundancy resolution. *International Workshop on the Algorithmic Foundations of Robotics*