Sampling-Based Motion Planning: A Comparative Review

Andreas Orthey and Constantinos Chamzas and Lydia E. Kavraki

Abstract

Sampling-based motion planning is one of the fundamental paradigms to generate robot motions, and a cornerstone of robotics research. This comparative review provides an up-to-date guideline and reference manual for the use of sampling-based motion planning algorithms. This includes a history of motion planning, an overview about the most successful planners, and a discussion on their properties. It is also shown how planners can handle special cases and how extensions of motion planning can be accommodated. To put sampling-based motion planning into a larger context, a discussion of alternative motion generation frameworks is presented which highlights their respective differences to sampling-based motion planning. Finally, a set of sampling-based motion planners are compared on 24 challenging planning problems. This evaluation gives insights into which planners perform well in which situations and where future research would be required. This comparative review thereby provides not only a useful reference manual for researchers in the field, but also a guideline for practitioners to make informed algorithmic decisions.

xxxxxx 0000. 00:1–25

1. Introduction

The last decades have seen remarkable progress in the field of robotics. An area of growth has been the development of sampling-based motion planning methods (1, 2, 3, 4), which have enabled applications such as robotics construction (5), multi-robot coordination (6), autonomous driving (7), industrial manufacturing (8), and protein folding (9).

Sampling-based motion planning (SBMP) is an approach to the problem of finding a motion for a robot to move from A to B. The distinguishing characteristic of SBMP is that it relies on sampling configurations (placements of the robot) in order to quickly find feasible and, in certain cases, optimal robot motions. Studies (10) have shown this to be one of the most efficient ways to solve problems with high numbers of degrees of freedom.

However, despite this success, incoming researchers and practitioners have currently no up-to-date guideline to use SBMP methods. In particular, there is currently no historical treatise recapping the development of the field until the present day. There exists also no categorization of planners summarizing the developments of the last decade. Moreover, it is often unclear how SBMP methods differ from alternative motion generation frameworks such as motion optimization, motion primitives, search-based planning or control-based planning. Finally, it is often unclear which planner class is best suited for a particular application area, and which planner performs best for a particular scenario.

To mitigate those gaps, this comparative review makes four core contributions:

1. an overview and brief history of SBMP methods, which spans the years 1979 to 2023,
2. an analysis and categorization of SBMP algorithms to give insights into different planners classes and extensions,
3. a discussion of how SBMP compares to alternative motion generation frameworks (e.g., (11, 12, 13, 14)), and its advantages and disadvantages,
4. a large-scale comparative evaluation on 24 scenarios, comparing several SBMP methods in terms of success, runtime, and optimality.

Those contributions make this comparative review an extensive guideline and reference manual to leverage the power of SBMP methods.

2. Motion Planning History and the Emergence of Sampling-based Methods

The history of motion planning begins around 1979 (15) and continues as an ever-growing research field into the present. This 40-year period can be divided into four eras. First, the pre-sampling era, where fundamental results were discovered and the problem was rigorously analyzed. Second, the sampling-advent era, where the first planners based on random sampling were discovered. Third, the sampling-consolidation era, where many improvements on sampling-based planners were made. Fourth, the optimality and learning era, where the first asymptotically-optimal planners were discovered and where learning algorithms became a focus of the community.

2.1. Pre-sampling Era (1979-1989)

The start of the motion planning era can be placed around 1979, when Lozano-Pérez (15) introduced the concept of the configuration space as a general framework to plan motions for arbitrary kinematic systems. This idea of planning through a configuration space (16) put the research field on a solid foundation and helped properly articulate the problem of

finding a path through the configuration space as the motion planning problem, sometimes also called the piano mover's problem (17).

After the motion planning problem was articulated, many researchers focused on the topic of computational complexity (17, 18, 19, 20). Fundamental results of that time were the proof that motion planning in configuration space is NP-hard (19, 20) and the development of the celebrated Canny algorithm which solves the problem in single exponential time (20). While the proposed motion planning algorithms have strong guarantees, they were not suited for practical, relevant problems due to their high computational complexity.

This was partially overcome by Khatib (21), who introduced the artificial potential-fields. This method uses attractive and repulsive artificial potential fields to drive the robot towards a desired goal while avoiding obstacles. Potential-fields were a popular method to control robots, with several extensions like the Laplacian potential field method (22), addition of circular directions (23), navigation functions (24), and numerical potential fields (25). While all those methods had tremendous influence on later algorithms like task-space control, they do sacrifice guarantees like completeness or optimality of the solution.

2.2. Sampling-advent (1990-1999)

The second era of motion planning comprises the years 1990 until 2000, where sampling-based algorithms were first pioneered and showed remarkable results in terms of efficiency of computation (26). This era roughly starts with the works of Barraquand (27), who improved the potential field approach using a Monte-Carlo random walking method which were instrumental for subsequent sampling-based planners. Sampling-based planners differed from other approaches by estimating the connectivity of the free configuration space through sampling. Despite not having an explicit representation of the configuration space, several of them were able to hold the property of probabilistic completeness, meaning that they will find a solution when time goes to infinity.

The most popular sampling-based approaches came in two categories. The first category were graph-based planners, like the probabilistic roadmap planner (PRM) (1), which randomly samples configurations, connects them to a graph, and uses targeted sampling to connect graphs. Several planning queries can then be answered using the same graph. The second category were tree-based planners, like the expansive-space trees (EST) (2, 28) and the rapidly-exploring random tree (RRT) algorithm (29, 3). They both grow a tree from a start configuration by randomly sampling configurations and connecting them to the tree till the goal is reached. The growth of sampling-based methods was starting (30).

2.3. Sampling-consolidation (2000-2009)

After several sampling-based planners were developed, it became clear that there are several areas where improvements could lead to another leap in terms of runtime. One of those areas were biased sampling functions, which do not sample uniformly the space, but bias samples towards certain areas. Several approaches were developed during this time, like sampling near or on the surface of obstacles (31), sampling inside narrow passages (32, 33), Gaussian sampling around current frontier states and obstacles (34), sampling restricted to workspace geometries (35) and workspace decompositions (36, 37), sampling on the medial axis of the environment (38), utility-based sampling to connect separate regions of roadmaps to each other (39), and sampling in areas that are deemed difficult (1). The dynamic-domain RRT (40, 41) extended tree nodes based on their estimated exploration ability.

2.4. Optimality and Learning period (2010-today)

Before the year 2010, sampling-based planning algorithms did not explicitly consider optimality. Instead, optimality was supposed to be a post-processing step, where a sampling-based planner provides a path which is then fed to an optimizer (42) for further improvement (43, 44, 45). In 2010, another breakthrough occured, where the star versions of PRM and RRT, PRM* and RRT* (4, 46), were developed. PRM* and RRT* are guaranteed to be asymptotically optimal, meaning they converge to the optimal solution in terms of path length at the limit, as the number of samples goes to infinity.

Besides achieving optimality guarantees, another important step was the integration of machine learning into motion planning. Robots often operate in similar environments solving similar motion planning problems. This motivated the use of past planning experiences to expedite the search in future problems. One approach of leveraging past experiences included retrieving a past solution that is similar to the current problem and repairing it. Past solutions are stored either in the form of path libraries (47), sparse roadmaps (48), or local obstacle roadmaps (49). Other approaches learn sampling distributions that can be used to bias the search of sampling based planners. Some methods learn sampling-distributions that are problem invariant (50) conditioned on the workspace description (51). More recent approaches based on deep learning can learn sampling distributions from past examples conditioned on workspace information, start, and goal information (52, 53). A recent review on learning for sampling-based planners summarizes those works (54).

State space: The set of all states uniquely describing a system plus additional structure like metrics, constraints, dynamics, or topology.

3. Motion Planning

The goal of motion planning is to develop algorithms to move mechanical systems (robots) from a start state to a goal region (55, 56, 57). A mechanical system consists of links and joints which can exist in different configurations or states based on the position or velocity of their joints. The set of all states of a system is called the *state space*[1]. The state-space is denoted by the letter X and its elements as x.

Not all states in the state space are physically feasible—they might violate a *constraint* (see Sec. 3.3). Constraints divide the state space X into the constraint-free region X_{free} and its complement $X \setminus X_{\text{free}}$. A *motion planning problem* is a tuple $(X_{\text{free}}, x_I, X_G)$, representing the task of finding a path, a continuous function $p : [0, 1] \to X_{\text{free}}$, from a start state $x_I \in X_{\text{free}}$ to a goal region $X_G \subseteq X_{\text{free}}$. The set of all feasible paths P is defined as the *path space* $P(X_{\text{free}}, x_I, X_G)$.

Motion planning problems can have several variations. The most important ones are

- **Path planning.** The geometrical problem ignoring the velocity, time, or dynamics of the system (57). This is often referred to as the piano mover's problem (17).
- **Kinodynamic planning.** Planning with a dynamical system and possible constraints on velocity, acceleration, or torque. A kinodynamic planning problem can be defined as a tuple $(X_{\text{free}}, x_I, X_G, f)$, where f are the dynamical equations (Sec. 4.5).
- **Optimal planning.** The problem of finding a global optimal path. An optimal path is a path p which minimizes a given cost functional $c : P \to \mathbb{R}_{\geq 0}$. An *optimal motion planning problem* is a tuple $(X_{\text{free}}, x_I, X_G, c)$, where the goal is to find a feasible path

[1] State space is used here as a general umbrella term to denote all spaces like configuration space, joint space, Cartesian space, parameter space, or phase space.

$p^\star$, such that $c(p^\star) = c^*$ and c^* is the minimum cost over the path space P.

The discussion that follows will focus on path planning, while kinodynamic planning is discussed in Sec. 4.5. Optimal planning is interleaved with the description of path planning and kinodynamic planning.

3.1. State Space Structure

The state space X needs to have additional structures. This includes it being a topological space (58), which is required to define the notion of a path and of path-connectedness. Most planners further assume that the state space is a *manifold*. A manifold is a topological space, which locally resembles an Euclidean space $\mathbb{R}^n$. This is an important assumption, because mechanical systems are naturally modelled by manifolds (59). Additional structures also include metric functions and constraints.

3.2. Metric Function

Most planners require a way to measure distances. This can be achieved by adding a *metric function* defined as $d : X \times X \to \mathbb{R}_{\geq 0}$. Given any elements $x, y, z \in X$, a metric function is characterized by the following three assumptions:

M.1 $d(x, y) = 0 \iff x = y$ (Identity of indiscernibles),
M.2 $d(x, y) = d(y, x)$ (Symmetry),
M.3 $d(x, y) \leq d(x, z) + d(z, y)$ (Triangle inequality).

If a metric cannot be defined on a problem, less-restrictive functions can be defined, which sacrifice one of the assumptions of a metric. The most important ones are:

- **Pseudometric**: Replace M.1 by the assumption $d(x, x) = 0$ for all $x \in X$ (60). An example is end-effector distance, which is zero for different inverse kinematic solutions.
- **Quasimetric**: Remove assumption on symmetry M.2 (61). An example are time-dependent state spaces, where the robot can only move forward in time (62).
- **Semimetric**: A metric invalidating the triangle inequality M.3 (63). A simple example are measurement errors violating the triangle inequality (64).

However, one has to be careful sacrificing metric properties, because some methods like nearest neighbor computations can depend on them. There exists also sampling-based planners that operate without a metric (65).

3.3. Constraint Functions

A constraint function codifies what a state has to satisfy to be considered feasible. Feasibility is problem-dependent and often involves avoiding obstacles, pushing an object, turning an object, or requiring that the robot stays close to a surface. Those tasks can be formalized using a constraint function $\phi : X \to \mathbb{R}$, which evaluates to less or equal to zero if a state is satisfied (or constraint-free), or to a value larger than zero otherwise. Most classical motion planning problems can be formulated using one of the following constraints.

- Collision constraint. Reject all states where robot links are in collision, either to other robot links (self-collisions), or to links in the environment. The constraint function

often only returns binary values, but can also be extended to return clearance or penetration depth.

- Joint limit constraint. Reject all states which violate joint limits on the robot. This might be a simple interval check for some actuators, but might also involve coupled joint checks where cables restrict the motion.
- Tool-center-point (Tcp) constraint. Reject states where the Tcp of the robot is outside certain limits. The Tcp is usually a predefined coordinate frame on the robot, where a tool is attached e.g., for spraying, painting, or welding.
- Kinodynamic constraint. Reject states which do not fulfil velocity, acceleration, or jerk limits on the robots joints. This is important for dynamical systems, where not all geometric paths have a valid velocity profile.

4. Sampling-based Motion Planning

Sampling-based motion planning is the idea of implicitly representing the state space through the use of a sampling function (Sec. 4.1). The sampling function generates a sequence of states which can be connected using a local planner (Sec. 4.2). To coordinate sampling and local planning, different planners have been developed, which are categorized in Sec. 4.3 either as tree-based planners (often used in single query problems), or graph-based planners (often used in multi-query problems). Over the years several general-purpose improvements have been proposed (Sec.4.3.1) that enhance different planner aspects. Besides performance improvements, there are many special-purpose sampling-based planners addressing variations of the canonical motion planning problem such as planning in unbounded spaces (Sec. 4.4.1), infeasible problems (Sec. 4.4.2), planning with kinodynamic constraints (Sec.4.5), and other motion planning extensions (Sec. 4.6).

4.1. Sampling Function

A sampling function generates an infinite sequence of elements of the state space as $S = \{s_1, s_2, \cdots\}$. For planners to offer guarantees, the sequence S is required to be dense in the state space X, which means that every point of X is arbitrarily close to a member of S.

Sampling functions are classified as unbiased or biased methods. An unbiased method, or uniform sampling, draws elements of the state space, whereby each outcome has an equal chance (66). Biased methods, however, change the probability distribution by biasing sampling towards interesting regions of the state space. While different biases can be used, sampling-based systems often favor one of the following three.

A first method is obstacle-based sampling (31). States close to an obstacle have a higher chance of being selected, i.e., there is a bias towards the boundary of the free state space X_{free}. Well known obstacle-based samplers are the Gaussian sampling method (34), and the bridge-based sampling method (33). This bias often improves planning in narrow passages (31), but also imposes an implicit bias on path length, which might interfere with other cost functionals like clearance.

A second method is clearance-based sampling (38). Those samplers prioritize samples which increase clearance, i.e., the distance between robot and environment. This can be achieved by sampling a feasible state, and making random steps to improve its clearance (67). Clearance-based sampling can mitigate execution uncertainty on real robots, but computing clearance queries is often expensive.

A third method is deterministic sampling (68, 69). Deterministic samplers reproduce the same sampling sequence in each run. Those sequences can be learned from similar environments to bias samples towards optimal paths (52), or they can achieve better distributions by minimizing the largest uncovered area (low-dispersion). Examples of low dispersion sequences are Halton sequences and Sukharev grids (56).

4.2. Local Planning

To find a continuous connected path, it is necessary to connect two state-space samples to each other and return a path segment (an edge) connecting them. This is accomplished using a *local planner*. A local planner is called local because the path segment often connects samples over a short distance using simple path segments (e.g., a straight line). It is not global in the sense that it does not address the complete global motion planning problem. A local planner is usually fast, but the produced path might not satisfy the constraints of the problem e.g., it might be in collision.

Depending on the category of the motion planning problem (Sec. 3) the local planner might need to satisfy additional constraints. In the case of optimal planning the produced path must be a lower-bound on the true solution cost. In the case of kinodynamic motion planning the differential constraints must be satisfied (see Sec. 4.5). Having efficient and optimal local planners is an active area of research with different methods leveraging local optimization (70) or learning (71) to improve their performance.

4.3. Categorization of Sampling-based Planners

Sampling-based planners can be classified into two main categories (55, 56, 57), *graph-based* (or roadmap-based) and *tree-based* planners. Algorithm 1 includes the BASIC-PRM (1) as a representative example of graph-based planners. A graph-based planner produces a graph by sampling (Sec. 4.1) constraint-free states and adding them to the graph (Line 4-5). Edges to nearest neighbors are added using a local planner (Sec. 4.2) (Line 6-10) to connect to the graph. The planner terminates if a planner terminate condition (PTC)[2] is fulfilled (Line 3). These planners are often called *multi-query*, because the graph can be reused for changing start states or goal regions. Examples include the probabilistic roadmap planner (PRM) (1, 4), and the sparse roadmap method (72).

A representative tree-based planner is BASIC-RRT (Algorithm 2) (3). Using a sam-

Algorithm 1 Basic-PRM

```
1:  procedure BASIC-PRM(x_I)
2:      G.addNode(x_I)
3:      while PTC is false do
4:          x_new ← valid sample from X_free
5:          G.addNode(x_new)
6:          N(x_new) ← K closest neighbors of x_new
7:          for each x_near ∈ N(x_new) do
8:              e ← Local plan x_new to x_near
9:              if e ∈ X_free and e ∉ G.edges() then
10:                 G.addEdge(e)
11:     return G
```

Algorithm 2 Basic-RRT

```
1:  procedure BASIC-RRT(x_I)
2:      T.addNode(x_I)
3:      while PTC is false do
4:          x_rand ← sample from X
5:          x_near ← nearest node in T to x_rand
6:          x_new ← Extend x_near towards x_rand
7:          e ← Local plan x_new to x_near
8:          if e ∈ X_free then
9:              T.addEdge(e)
10:     return T
```

[2]The planner terminate condition can for example be a successful solution, a timeout, or a maximum number of iterations.

pling function, a random sample from the state space is chosen (Line 4). This random sample is used to extend from the nearest state in the tree (Line 5-7), that is to generate a path starting from the nearest state to the random sample with a local planner. If the resulting edge is constraint-free, it is added to the tree (Line 8-9). These planners are often called *single-query*, because the trees need to be recomputed for different start states or goal regions. Examples include the rapidly-exploring random trees (RRT) planner (3, 4), the expansive-space trees (EST) planner (2), the lower-bound trees RRT (LBT-RRT) (73), and the fast-marching trees planner (FMT) (74).

4.3.1. General-purpose Planner Improvements. Planner efficiency can improve by adding or improving one of the following components. Efficiency here can mean reducing memory footprint, decreasing runtime, decreasing number of samples, or improving path cost.

Lazy Checking. A lazy version of a planner (28, 75, 76, 77) ignores edge constraint checking during planning. This reduces memory footprint and speeds up runtime. Once a solution has been found, the edges are checked for constraint violations. In the case of a constraint violation, the edge is removed from the tree or graph and planning is continued. This method can be seen as a constraint relaxation method, where a simplified problem is solved first, and this information is leveraged to solve the original problem (78).

Bidirectionality. Most tree-based planners can be extended to plan not only with one, but two or more trees. An example is the bidirectional RRT planner (3) which alternates between extending two trees grown from start and goal, respectively. On each successful extension, a connection between the nearest states in the trees are tried. This approach can also be extended to optimal versions (79). In general, this decreases runtime significantly.

Sparsity. Most planners can also be made sparse. A sparse planner often ignores samples which are inside of a visibility radius of a given node in the graph or tree (80). Another way to ensure sparsity is to only keep the best cost-to-come states in certain regions of the state space (81). This is a good way to reduce the memory footprint of the planner.

Optimality. In the pioneering work by Karaman and Frazzoli (4), the authors show that many planners can be adapted to make them converge to the global optimal solution (asymptotic optimality). This involves having an adaptive nearest neighbors radius for graph-based planners (46), or using a tree-rewiring operation after each sampling iteration (82). This approach usually improves the cost of paths significantly.

Admissible Heuristics. Admissible heuristics are lower bound estimates on the cost to reach a goal (83). One important category are *informed sets* (84), which describe all states which can improve the solution quality. Examples include the batch-informed trees planner (BIT*) (85), and the advanced-informed trees planner (AIT*) (86). Another category are constraint relaxations (78), where (multiple) levels of projections are used to simplify the problem, as in the KPIECE method (87, 88), the quotient-space rapidly-exploring random tree planner (QRRT*) (78), and the hierarchical fast marching tree planner (HFMT*) (89). A good admissible heuristic can decrease the planner runtime significantly without sacrificing completeness or optimality.

Parameter-Tuning. Most planners have parameters e.g., the number of nearest neighbors K in Basic-PRM (Algorithm 1), that need to be chosen before planning. In certain problem scenarios these parameters can significantly affect planning performance. Although planning frameworks such as OMPL (90) often offer reasonable defaults, choosing an appropriate set of hyperparameters is considered an open research problem. Some recent works have investigated bayesian optimization to automatically choose these parameters (91, 92).

4.3.2. Planner Properties. Sampling-based planners set themselves apart from competing motion generation frameworks (see Sec. 5) by providing desirable guarantees. One guarantee is *probabilistic completeness*. Probabilistic completeness states that a planner will find a solution path if one exists, when time goes to infinity.

Probabilistic Completeness: The planner will find a solution if one exists, when time goes to infinity.

Another important guarantee is *asymptotic optimality* (4, 93). Asymptotic optimality states that a planner will find the global optimal solution path when time goes to infinity. Algorithms with this property, like RRT*, PRM* (4), or BIT* (94), are able to continuously improve the solution cost until they eventually converge to the global optimal solution (4). A slightly weaker notion is *asymptotic near-optimality*. This is a variant of asymptotic optimality stating that a planner will find a solution when time goes to infinity whereby the solution cost is ϵ-near to the cost of the optimal solution. Some planners like SPARS (72) provide this weaker notion to trade-off memory consumption with optimality guarantees.

Asymptotic Optimality: The planner will find the global optimal solution when time goes to infinity.

4.4. Special Cases of Motion Planning

While several sampling-based planners are general enough to solve any motion planning problem, there are some special cases, which require special care.

4.4.1. Unbounded Space. Some problems, like planning in space-time, require sampling of an unbounded state space (see Fig. 1). However, most sampling sequences require a bounded space to generate dense samples. To resolve this discrepancy, one option is to use adaptive goal regions (62), where lower and upper bounds are shifted to keep asymptotic optimality while having a bounded region to enable sampling. Another option is to select existing nodes using a selection-extension scheme, where nodes are selected at random or based on their utility for expansion (40). Those nodes can be extended into random directions or via dynamics-driven propagation functions (81).

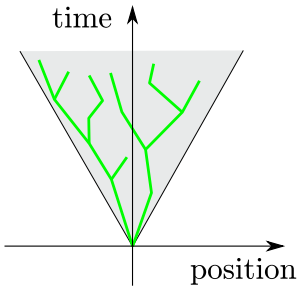

Figure 1: Unbounded time-dependent state space, with no natural time limit.

4.4.2. Infeasible Problems. While sampling-based planners usually provide probabilistic completeness guarantees, they often cannot deal with infeasible planning problems (see Fig. 2). To handle infeasible problems, there are different methods which can be applied. One method is based on sparse roadmaps (80, 72). A sparse roadmap has a given visibility radius, such that new samples inside the visibility radius are rejected, and the number of samples added to the roadmap goes to zero if time goes to infinity. If no samples can be added for a certain period, a probabilistic estimate of infeasibility can be given (80, 95).

Another method is based on infeasibility certificates (96, 97, 98). Such methods can tackle lower-dimensional problems, where samples are used to create a closed hull around the start state or around the goal region to verify that a problem is infeasible (98).

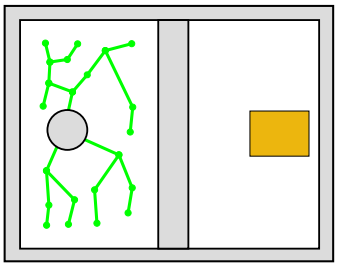

Figure 2: Infeasible problem, where a disk robot (grey) can plan motions (green), but cannot reach a goal (orange) due to obstacles (grey).

4.5. Kinodynamic Motion Planning

In addition to kinematic constraints, many realistic systems have to satisfy dynamics and constraints on velocity, acceleration, or torques. This is known as kinodynamic motion planning (56). The dynamics of kinodynamic systems can be expressed as:

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

whereby $\dot{x}$ is the derivative of x, $u \in U$ the controls of the system, U the space of applicable controls, and f the dynamical systems equation. This dynamics equation can be seen as imposing a constraint on the allowable paths the system can take. There are two main ways of how planners can handle dynamics.

Steering method. In this approach, given an initial state x_1 and a target state x_2, the dynamics f are used to analytically or numerically compute a control u and time t which respects the dynamics and moves the state from x_1 to x_2. This simplifies planning, but steering might be difficult and costly to compute. Examples of systems with analytical steering methods are Dubin's car (99) and the Reeds-Sheep car (100). In general, a two-point boundary value problem (BVP) between two states needs to be solved. The solution of a BVP provides a local planner which can be used in any geometric planning algorithm to eventually solve the kinodynamic problem.

Forward propagation The second approach uses forward propagation, whereby an initial state x_1 and a control u is used, and the system is forward propagated for a specified time t to compute the next state x_2. This is advantageous since the dynamics can be treated as a black box. Kinodynamic planners like Kinodynamic-RRT (3) can solve those problems by using random control sampling to propagate states forward. Asymptotic optimality can be achieved using the meta planner AO-RRT (101) which relies on the Kinodynamic RRT to produce feasible solutions in a combined cost-state space. A kinodynamic planner which combines asymptotic near-optimality and efficiency is the stable sparse RRT (SST) (81), which grows a tree in state space while using pruning to maintain only the locally best solutions (67). An asymptotic optimal version (SST*) can be achieved by iteratively decreasing the pruning resolution (81).

4.6. Extensions Solved by Sampling-based Motion Planning

An extension of motion planning is defined as a problem which imposes additional structures onto the state space. This additional structure can either be dictated by the problem itself, such as contact constraints, or partial-observability. Or the structure is added intentionally to improve planner performance, like projections, or differentiability. This section gives a non-exhaustive list of additional structures, and how sampling-based planners can be extended to accommodate them.

4.6.1. Projection-based Motion Planning. Many high-dimensional problems can often be simplified by introducing projections (78, 88, 89). Projections allow planning over different levels of abstraction. There are different ways to plan with projections, either by using them as biased samplers (88), or by using projections to adjust and guide the sampling. If projections are admissible, i.e., solving a simplified problem is a necessary conditions on the original problem, properties like asymptotic optimality can be guaranteed (78, 89). While projections usually speed up planning significantly (78), it is often difficult to define them for a new problem domain.

4.6.2. Differentiable Motion Planning. Most functions used in motion planning, like costs, goals, or constraints, are often differentiable. Exploiting those functions is studied under the topic of differentiable motion planning (102). While most sampling-based planner avoid differentiable information, there is evidence that differentiable information can be useful to converge faster to optimal solutions (103). Differentiable planners need to carefully weigh when and if computing differentiable information is useful. Recent work in this direction combines optimization with sampling by adding graph optimization (104), improving approximately valid paths (103), or optimizing the steering function (70).

4.6.3. Planning with Manifold Constraints. Most constraints in classical motion planning can readily be sampled to construct satisfiable state space elements. However, complex constraints, like contacts between robot and environment, often introduce regions in the state space having zero measure, which have zero chance to be sampled. Dealing with such constraints is called motion planning with manifold constraints (105). There are different ways of how to deal with those constraints. For example, the volume of the constraints can be relaxed to simplify the problem, or random samples can be projected back to the constraints (105, 106).

4.6.4. Motion planning in Dynamic Environments. Often, robots have to deal with obstacles which might move, appear, or disappear. Dealing with such obstacles is studied under the topic of *planning in dynamic environments* (107). Since successful plans might get invalidated, specialized planners are required. Examples include RRTX (108), which updates a goal-centered search tree on the fly, and precomputed roadmaps to quickly recompute solution paths (8).

4.6.5. Belief Space Planning. Many realistic robots do not have access to a fully observable environment, but need to discover the world using sensors. This leads to the problem of *planning in partially-observable environments* (109). To solve those scenarios the belief-space can be sampled (110), which contains hypotheses about the world plus robot states. Since the space of hypotheses might get large, it is important to properly simplify those spaces to make them searchable (111).

4.6.6. Planning in Force Fields. Often, robots need to handle external forces arising from gravity, friction, or wind. Sampling-based motion planning frameworks can integrate forces as vector fields on the state space of the robot (112). Planning methods in this area mostly focus either on handling wind disturbances in Unmanned Aerial vehicles (UAV) (113), or handling water draft in Autonomous Underwater Vehicles (AUV) (114, 115).

5. Competing Motion Generation Frameworks

Sampling-based motion planning is a framework to generate motions for arbitrary mechanical systems. However, there are competing frameworks to generate motions. Those frameworks are briefly surveyed and advantages and disadvantages are listed relative to sampling-based motion planning.

5.1. Motion Optimization

A framework with complementary strengths to sampling-based planning is motion optimization. The idea behind motion-optimization is to formulate motion generation as an optimization problem (116), and often focus on improving an existing path.

A distinction can be made between gradient-free and gradient-based methods. Gradient-free methods include shortcutting (42) or hybridizing[3] (44) to improve path length. Interpolation of splines (117) can be used to improve path smoothness. Gradient-based methods often make the cost, the goal, and the constraints differentiable, which allows them to quickly find low-cost paths. Examples include general-purpose optimization frameworks like CHOMP (118), TrajOpt (119) KOMO (12), GPMP(102), to optimize motions conditioned on arbitrary task requirements.

Advantages. Since most optimization-based methods use second-order information, they can quickly converge to low-cost solutions. Most algorithms can also handle information to push robots out of collisions, which makes those methods applicable even when a returned path is in collision (118).

Disadvantages. Optimization-based methods usually can only find locally optimal solutions, and lack guarantees on completeness or optimality. If the starting path is not constraint-free, one may end up with an invalid path—even if a feasible solution exists.

5.2. Motion Primitives

Motion primitives (120, 121, 122, 123) are predefined or learned motions to accomplish a certain task. They can be defined as dynamical systems in the state space, which provide a vector field along which the robot can move to reach a target, fulfill a task, or avoid an obstacle. Those primitives are loosely based on insights from neuroscience (124), which have shown that animals are often moving by composing sets of motion primitives for reaching or walking motions (125).

To equip robots with motion primitives, they can either be learned or predefined based on task requirements like avoiding obstacles or reaching a target pose. Motion primitive frameworks like Riemannian motion policies (13), are able to compose complex motions by combining several simple task policies, whereby a policy is a function telling the robot where to move from anywhere in the state space.

Advantages. Policies can be used for reactive planning, where obstacles can be avoided in realtime (126). This can be a good choice for problems where feedback is crucial, like playing table tennis (122).

Disadvantages. Motion primitives usually do not provide guarantees on completeness or optimality. Finding primitives is often difficult and task-dependent. To make the methods work, it is often necessary to fine-tune the primitives and control policies.

5.3. Search-based Planning

Search-based planning (127, 11, 128) differs from sampling-based motion planning by imposing a grid onto the state space. By connecting neighboring states in this grid, a state space graph can be constructed. Based on this graph, search-based planners like A* (129)

[3]Hybridizing refers to the operation of combining a set of (possibly high-cost) input paths to generate a single (low-cost) output path.

can find optimal motions with respect to the resolution of the grid. This is an efficient method to quickly find solutions in lower-dimensional state spaces (130).

Advantages. Variants of A*-like algorithms can directly be used, which are guaranteed to solve a problem to the optimal solution—w.r.t. the resolution of the graph. Open-source software is available in the search based planning library (SBPL)[4].

Disadvantages. Only resolution-completeness can be guaranteed depending on the grid resolution. If the resolution is too fine, planning time might be exceptionally expensive. If the resolution is too coarse, narrow passages cannot be found or traversed. Search grids are also difficult to employ in high-dimensions due to the curse of dimensionality.

5.4. Control-based Planning

Control algorithms are useful to drive a robot towards a desired goal state. An example is the Proportional–integral–derivative (PID) controller (131), which uses current state information to compute the next input to the system. This can be combined with repulsive forces to push the system away from obstacle regions, as in the artificial potential fields approach (21). More global approaches are the Linear–quadratic regulator (LQR) method (132) which optimizes a (locally) optimal path, and the Model predictive control (MPC) method (133) which can find optimal path segments over a receding horizon. Controllers can often also be chained together to provide some form of guarantee, as in the funnels approach (134), where multiple local controllers cover the state space.

Controllers can also be learned using Reinforcement Learning (135) approaches, where the learned controller (policy) tries to maximize the given reward signal. A reinforcement learning (RL) algorithm improves itself by updating an underlying value function, which is a measure for how desirable states are depending on the reward signal. Over time, learning algorithms like Q-learning (136) will eventually converge to the optimal policy. For long-horizon plans, controllers are often combined with planning methods from the previous sections which compute a reference trajectory which is subsequently given to the controller.

Advantages. The above approaches achieve fast computation of locally optimal paths, whereby most controllers are reactive and simple to implement. Additionally, a controller can can continuously incorporate execution feedback.

Disadvantages. Controllers might get stuck in local minima, or return a sub-optimal path. There is often no guarantee on either completeness or optimality. If used inside a planning framework, computation time might become the bottleneck.

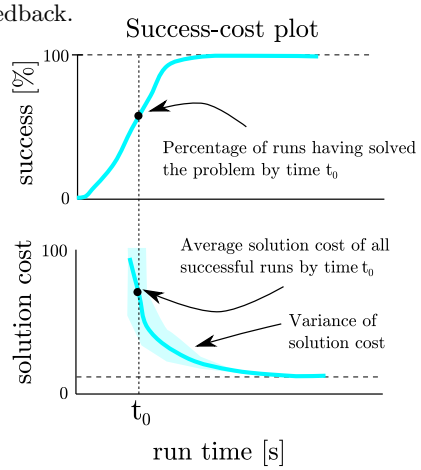

Figure 3: Success-cost plot explanation.

6. Comparative Evaluations

To reveal the relative performance of planning algorithms, several motion planners from the open motion planning library (OMPL)[5] (90, 137) are evaluated on a set

[4] `https://github.com/sbpl/sbpl`

[5] `https://github.com/ompl/ompl` and `https://ompl.kavrakilab.org`

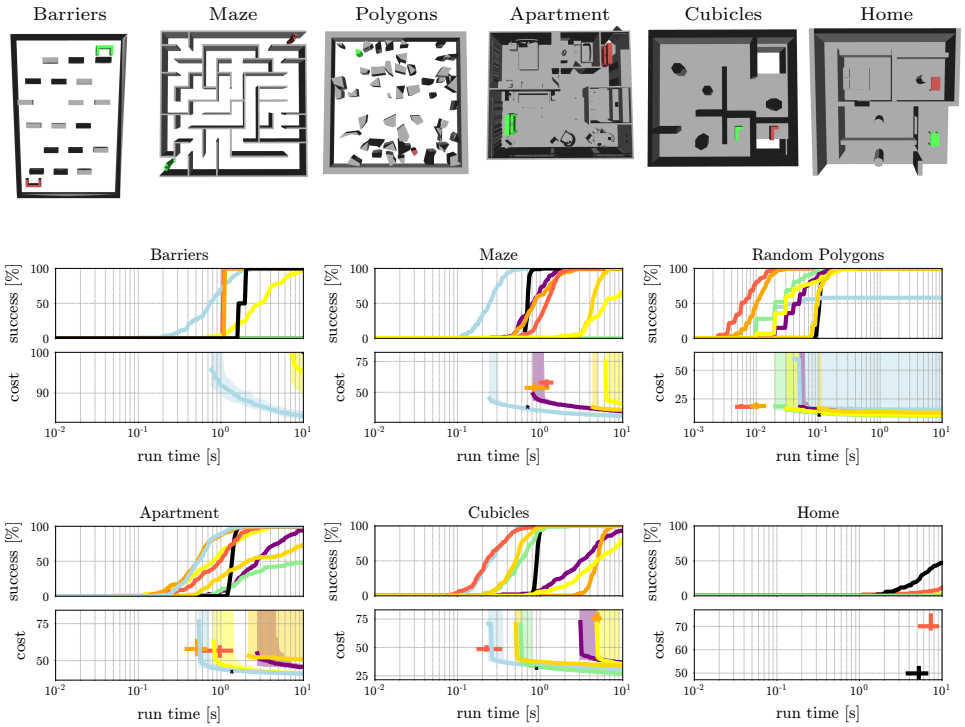

Figure 4: Classical Experiments with planners RRT-Connect, PRM, RRT*, FMT, EST, LBTRRT, AIT*, and BIT*.

of 24 scenarios. For each scenario, a set of applicable planners is selected and run for 100 runs with a scenario-dependent timeout up to 300s. For all experiments the default OMPL parameters of the planners were used. Changing those parameters could potentially influence the results (10), but the size of the experimental dataset prevents fine tuning. As hardware, a 4-core, 8GB RAM laptop running Ubuntu 16.04 is used. The results are shown as success-cost plots (see Fig. 3), with time on the x-axis, success rate and solution cost[6] on the y-axis. In the case of a non-optimizing planner, the average cost of the first solution is displayed as a single cross. When a color is not displayed, this means that the planner was not able to find any solution paths.

6.1. Classical Experiments

This set of experiments includes classical motion planning problems, which frequently appear in the motion planning literature. In each scenario, the task is to move a rigid body from a start to a goal region. The scenarios are

1. **Barriers**: A 3-DOF problem consisting of a horseshoe-like robot having to traverse a rectangle with several obstacles.
2. **Maze**: A 3-DOF problem, where a small disk-like robot has to traverse a maze.

[6]Solution cost is in all cases path length if not explicitly mentioned.

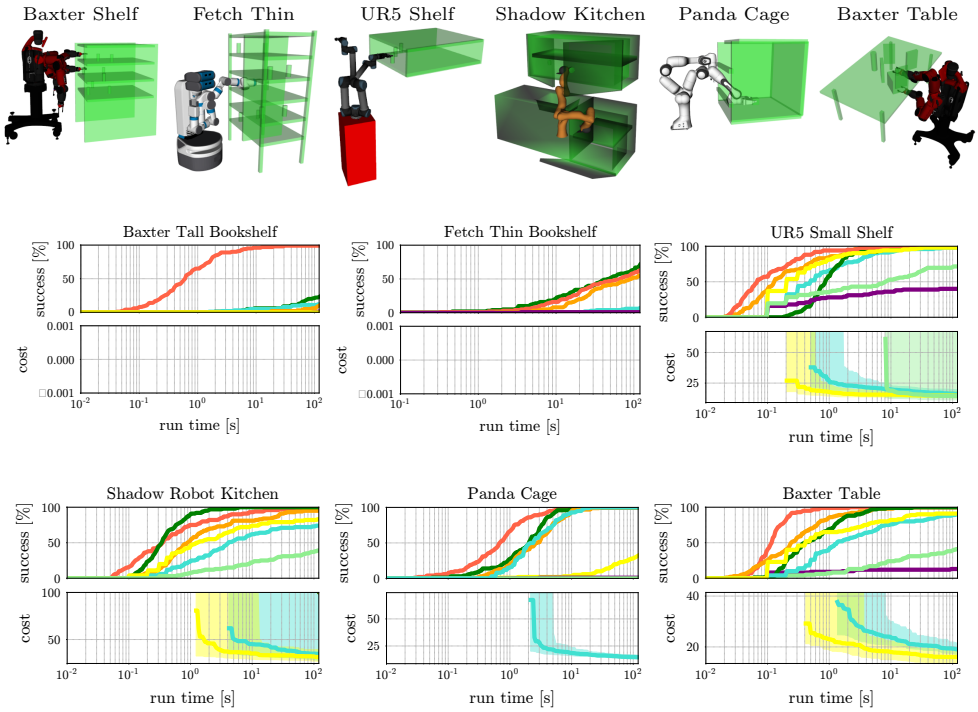

Figure 5: Manipulation experiments using planners RRTConnect, PRM, PRM*, RRT*, EST, KPIECE, and AIT*.

3. **Random polygons**: A 3-DOF problem, where a a disk-like robot has to traverse a room with random polygonal obstacles.
4. **Piano movers problem**: A 6-DOF problem, with a piano moving across a room.
5. **Cubicles**: A 6-DOF problem with a floating L-shape which has to traverse a cubicle.
6. **Apartment**: A 6-DOF problem, where a table moves through an apartment with furniture.

Fig. 4 shows the results. It can be seen that BIT*, EST, FMT and RRT-Connect perform well in terms of success rate on 4 out of 6 scenarios. FMT is slightly slower than the other planners, but is the only one solving the Home scenario in 50% of the cases. In terms of cost, BIT* has the best convergence in 5 out of 6 scenarios. Graph-based planners like PRM take longer to solve those problems. However, this performance is offset by the reusability of graphs for future planning queries. This evaluation underlines a common observation among practitioners, that there is no single planner that has the best performance across all scenarios. Choosing a suitable planner depends significantly on the robot and environment.

6.2. Manipulation Experiments

This set of experiments includes problems from the MotionBenchMaker dataset (53). In each problem the position of objects is varied relative to the robots.

1. **Ur5 Small Shelf**: 6-DOF problem where the UR5 robot starting outside a small shelf

reaches inside the shelf to pick up a cylindrical object.

2. **Franka-Emika cage**: 7-DOF problem where the FRANKA-EMIKA robot starting outside a caged-box is reaching inside to pick up a cube.
3. **Baxter table**: 7-DOF problem where the BAXTER robot starting with the arm under the table is reaching to pick a cylindrical object on a cluttered table.
4. **Fetch narrow bookshelf**: 8-DOF problem where the FETCH robot is starting from the home (stow) position and reaches for a cylindrical object deep in this narrow bookshelf.
5. **Baxter large bookshelf**: 14-DOF problem where the BAXTER robot s starting outside the shelf reaches to pick two objects with both arms.
6. **Shadow-Kuka kitchen**: 31-DOF problem where a shadowhand mounted on a kuka arm starts inside the dish-washer and reaches inside the kitchen's shelf.

As shown in Fig. 5, RRT-Connect has the best overall success rate and is the only planner reaching 100% in 5 out of 6 scenarios. Only KPIECE is able to reach 100% quicker than RRTConnect in 1 out of 6 scenarios. In terms of cost, PRM* converges quickly to a low-cost solution in 4, while AIT* in 3 out of 6 scenarios. RRT* only finds a low-cost solution in 1 scenario.

6.3. Narrow Passage Experiments

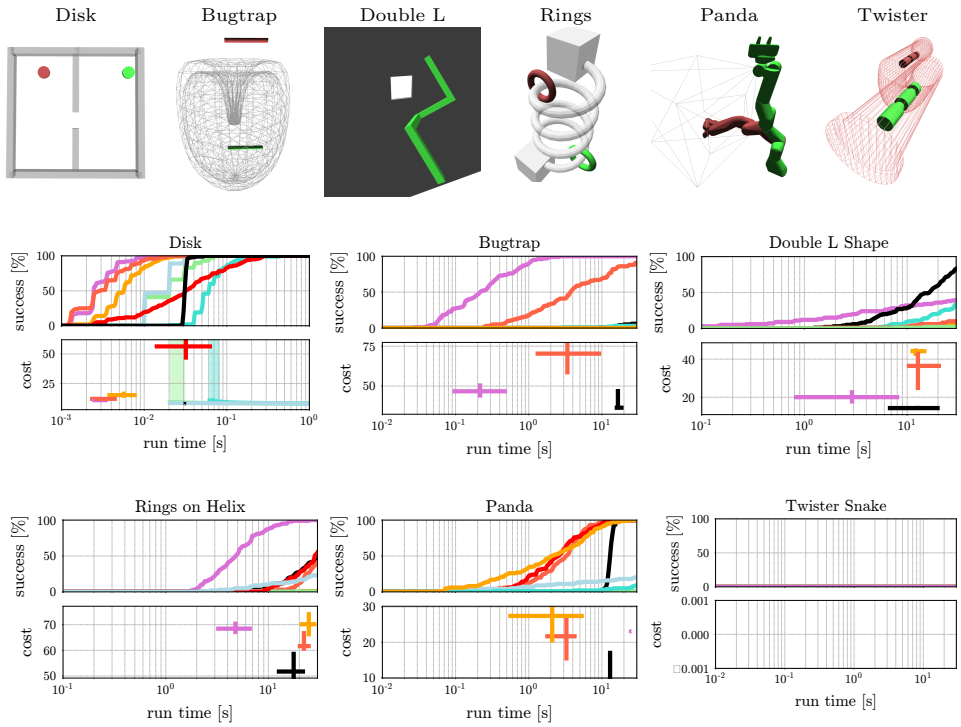

Figure 6: Limitations Experiments. Using planners RRT-Connect, PRM*, RRT*, KPIECE, FMT, EST, TRRT and BIT*.

This set of experiments includes six different types of narrow passages. Narrow passages

are difficult to solve for sampling-based motion planners, because the passages constitute bottlenecks, i.e. regions of the state space, which have near-zero measure, and near-zero probability to sample states (138).

1. **2d-Disk room**: A 2-DOF problem where a small disk moves through a slit in a wall.
2. **Bugtrap**: A 6-DOF problem, where a cylindrical object (the bug) escapes a trap.
3. **Double L-shape**: A 6-DOF problem, where an object consistent of two L-shapes has to traverse a square hole in a wall.
4. **Rings on a helix**: A 6-DOF problem, where a ring has to move along a helix.
5. **Franca-Emica box**: A 7-DOF problem, where a Franca-Emica robot has to move its endeffector inside of a box.
6. **Twister snake**: A 10-DOF problem, where snake-like robot traverses a twisted pipe.

In terms of success rate, the best planners are RRT-Connect and TRRT, which can solve 3 scenarios to almost 100% success (see Fig. 6). While optimal planners can only find low-cost solutions in one scenario, is is noteworthy to mention that FMT can find low-cost solutions in five out of six scenarios. The twister scenario was not solved by any planner.

6.4. Extension Experiments

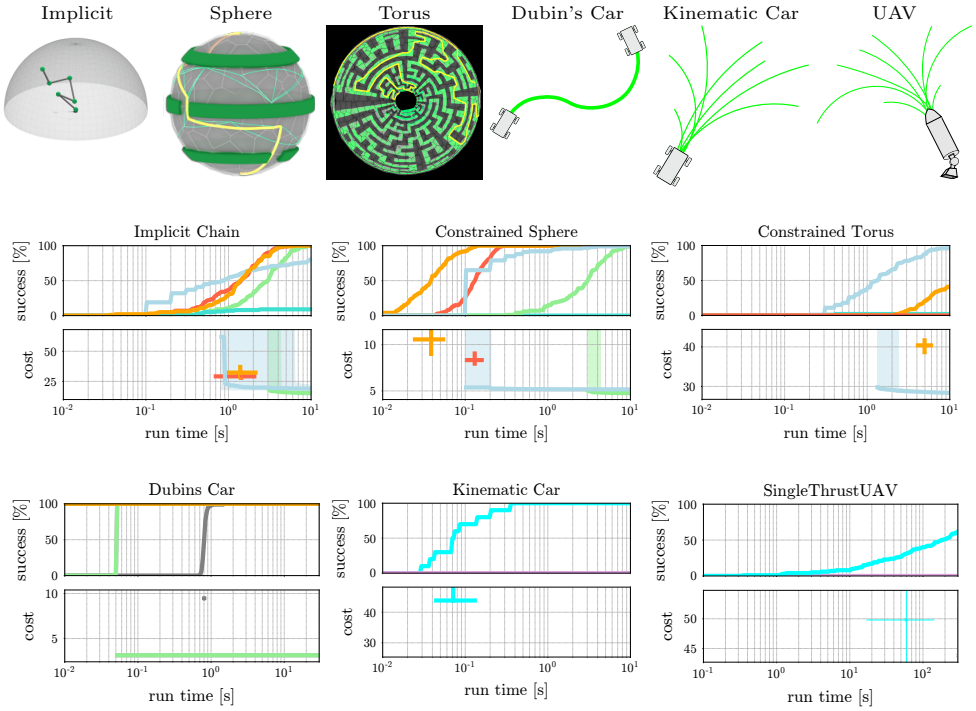

Figure 7: Extension Experiments. For the geometric experiments, the planners are RRT-Connect, PRM, EST, BIT*, RRT*, and FMT. For the dynamic system experiments, the planners are SST*, Kinodynamic RRT

The extension experiments include contact-constraints, where robots are restricted to move along surfaces, and scenarios, where a dynamic model is used to actuate the robot.

1. **Implicit chain**: A 5-DOF problem where the end-effector of an articulated chain is bounded to the surface of a sphere (139).
2. **Constrained sphere**: A 3-DOF point robot which is bounded to the surface of a sphere with obstacles (139).
3. **Maze on Torus**: A 3-DOF point robot which is bounded to the surface of a torus plus a maze-like obstacle (139).
4. **Dubin's car**: A 3-DOF simple car model which can drive forwards at a constant speed (using a steering function).
5. **Dynamic car**: A 3-DOF dynamic model of a car with a 2-DOF action space, which can accelerate and steer (using a propagator function).
6. **Single-thruster UAV**: A 9-DOF dynamic model of an unmanned aerial vehicle (UAV), which has a single actuated thruster which can be pivoted.

Fig. 7 shows the result. Using the constraint-based framework (139), EST, BIT*, and RRT-Connect can solve two scenarios to 100% , while BIT* finds low-cost solutions in three. In the dynamic cases, only Kinodynamic-RRT is able to find solutions, while the optimal planner SST* cannot solve any of the scenarios.

7. Discussion

This comparative review provided a reference manual for using sampling-based motion planning algorithms. A large-scale evaluation of different planners showed which ones perform best on 24 challenging scenarios. The results indicate that planning algorithms can successfully solve a broad class of problems, even with narrow passages, constraints, or dynamics. However, there is not a single planner that performs best across all problems. Due to space limitations, a topic not covered in these experiments was the choice of hyper-parameters of sampling-based planners. Hyper-parameters, however, can drastically affect the performance of a motion planning problem. This is an active area of research and we refer the reader to (91) for a starting point that discusses hyper-parameter tuning for sampling-based planners. Apart from the comparative evaluations, this review provided a comprehensive overview about state space structures, categories of sampling-based planning algorithms, motion planning extensions, and a comparison to alternative motion generation frameworks. This should give researchers and practitioners the tools to make better decisions about which sampling-based planners to use in a specific motion planning scenario.

8. Other Reviews of Interest

In the last decades, researchers in sampling-based motion planning have published several comprehensive review papers. An early account of the history of the field is found in the treatise by Latombe (26). Focusing more on sampling-based approaches, the work by Tsianos et al. (140) discusses developments of the field during the early 2000's. A more comprehensive resource is the work by Elbanhawi and Simic (141) which provides an overview of different planners, together with a set of general primitives for all planners.

There is also an ever-growing list of review papers focusing on a specific variant of motion planning or on a specific application field. A comprehensive overview of asymptotically-optimal planners is a recent survey by Gammell and Strub (93). Similarly, but with a focus on heuristic approaches, is the work by Mac et al. (142). Excellent overviews exist also on motion planning inspired methods for molecular simulations (143, 144), where different

required extensions are discussed. For the field of Unmanned Aerial Vehicles (UAVs), the work by Goerzen et al. (145) provides an overview and discusses more advanced planning problems like surveillance and reconnaissance. Learning and sampling-based planning is discussed in (54). Excellent reviews for the role of motion planning in task planning can be found in Garrett et al. (146), and Kingston et al. (105), respectively.

While all these works provide good overviews about sampling-based planning and specific areas, this review provides a broader guide for sampling-based motion planning. This review is also more application-oriented, in that it also shows the relative performance of popular planners on different problem areas.

ACKNOWLEDGMENTS

Work on this article by Lydia E. Kavraki is supported in part by NSF RI 2008720.

LITERATURE CITED

1. Kavraki LE, Svestka P, Latombe JC, Overmars MH. 1996. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE Transactions on Robotics* 12(4):566–580
2. Hsu D, Latombe JC, Motwani R. 1999. Path planning in expansive configuration spaces. *International Journal of Computational Geometry and Applications* 9(4-5):495–512
3. Kuffner JJ, LaValle SM. 2000. *RRT-connect: An efficient approach to single-query path planning*. In *IEEE International Conference on Robotics and Automation*, vol. 2, pp. 995–1001
4. Karaman S, Frazzoli E. 2011. Sampling-based algorithms for optimal motion planning. *International Journal of Robotics Research* 30(7):846–894
5. Hartmann VN, Orthey A, Driess D, Oguz OS, Toussaint M. 2023. Long-horizon multi-robot rearrangement planning for construction assembly. *IEEE Transactions on Robotics* 39(1):239–252
6. Hönig W, Preiss JA, Kumar TKS, Sukhatme GS, Ayanian N. 2018. Trajectory planning for quadrotor swarms. *IEEE Transactions on Robotics* 34(4):856–869
7. Claussmann L, Revilloud M, Gruyer D, Glaser S. 2019. A review of motion planning for highway autonomous driving. *IEEE Transactions on Intelligent Transportation Systems* 21(5):1826–1848
8. Murray S, Floyd-Jones W, Qi Y, Sorin DJ, Konidaris GD. 2016. *Robot Motion Planning on a Chip*. In *Robotics: Science and Systems*, vol. 6
9. Al-Bluwi I, Siméon T, Cortés J. 2012. Motion planning algorithms for molecular simulations: A survey. *Computer Science Review* 6(4):125–143
10. Chamzas C, Quintero-Peña C, Kingston Z, Orthey A, Rakita D, et al. 2022. Motionbenchmaker: A tool to generate and benchmark motion planning datasets. *IEEE Robotics and Automation Letters* 7(2):882–889
11. Cohen BJ, Chitta S, Likhachev M. 2010. *Search-based planning for manipulation with motion primitives*. In *IEEE International Conference on Robotics and Automation*, pp. 2902–2908
12. Toussaint M. 2017. A tutorial on newton methods for constrained trajectory optimization and relations to slam, gaussian process smoothing, optimal control, and probabilistic inference. In *Geometric and Numerical Foundations of Movements*, ed. JP Laumond, N Mansard, JB Lasserre, pp. 361–392. Cham: Springer International Publishing
13. Cheng CA, Mukadam M, Issac J, Birchfield S, Fox D, et al. 2021. Rmpflow: A geometric framework for generation of multitask motion policies. *IEEE Transactions on Automation Science and Engineering* 18(3):968–987
14. Kober J, Bagnell JA, Peters J. 2013. Reinforcement learning in robotics: A survey. *International Journal of Robotics Research* 32(11):1238–1274

15. Lozano-Pérez T, Wesley MA. 1979. An algorithm for planning collision-free paths among polyhedral obstacles. *Communications of the ACM* 22(10):560–570
16. Lozano-Perez T. 1983. Spatial planning: A configuration space approach. *IEEE Transactions on Computers* 2(C-32):108–120
17. Schwartz JT, Sharir M. 1983. On the "piano movers" problem. II. General techniques for computing topological properties of real algebraic manifolds. *Advances in Applied Mathematics* 4(3):298–351
18. Reif JH. 1979. *Complexity of the mover's problem and generalizations*. In *Conference on Foundations of Computer Science*, pp. 421–427
19. Canny J, Reif J. 1987. *New lower bound techniques for robot motion planning problems*. In *28th Annual Symposium on Foundations of Computer Science (sfcs 1987)*, pp. 49–60. IEEE
20. Canny J. 1988. *The complexity of robot motion planning*. MIT press
21. Khatib O. 1986. Real-time obstacle avoidance for manipulators and mobile robots. In *Autonomous Robot Vehicles*. Springer
22. Sato K. 1992. Deadlock-free motion planning using the laplace potential field. *Advanced Robotics* 7(5):449–461
23. Koditschek D. 1987. *Exact robot navigation by means of potential functions: Some topological considerations*. In *IEEE International Conference on Robotics and Automation*, vol. 4, pp. 1–6
24. Koditschek DE, Rimon E. 1990. Robot navigation functions on manifolds with boundary. *Advances in Applied Mathematics* 11(4):412–442
25. Barraquand J, Langlois B, Latombe JC. 1992. Numerical potential field techniques for robot path planning. *IEEE Transactions on Systems, Man, and Cybernetics* 22(2):224–241
26. Latombe JC. 1999. Motion planning: A journey of robots, molecules, digital actors, and other artifacts. *International Journal of Robotics Research* 18(11):1119–1128
27. Barraquand J, Latombe JC. 1991. Robot motion planning: A distributed representation approach. *International Journal of Robotics Research* 10(6):628–649
28. Sánchez G, Latombe JC. 2003. A single-query bi-directional probabilistic roadmap planner with lazy collision checking. In *International Journal of Robotics Research*. Springer
29. Lavalle SM. 1998. Rapidly-exploring random trees: A new tool for path planning. Tech. rep., Iowa State University
30. Barraquand J, Kavraki L, Latombe JC, Li TY, Motwani R, Raghavan P. 1996. *A random sampling scheme for path planning*. In *Robotics Research: The Seventh International Symposium*, pp. 249–264. Springer
31. Amato NM, Bayazit OB, Dale LK, Jones C, Vallejo D. 1998. OBPRM: An obstacle-based PRM for 3D workspaces. In *Robotics: The Algorithmic Perspective*, ed. PK Agarwal, LE Kavraki, MT Mason, pp. 155–168. CRC Press
32. Hsu D, Kavraki LE, Latombe JC, Motwani R, Sorkin S. 1998. On finding narrow passages with probabilistic roadmap planners. In *Robotics: The Algorithmic Perspective*, ed. PK Agarwal, LE Kavraki, MT Mason, pp. 141–154. CRC Press
33. Hsu D, Jiang T, Reif J, Sun Z. 2003. *The bridge test for sampling narrow passages with probabilistic roadmap planners*. In *IEEE International Conference on Robotics and Automation*, vol. 3, pp. 4420–4426
34. Boor V, Overmars MH, Van Der Stappen AF. 1999. *The Gaussian sampling strategy for probabilistic roadmap planners*. In *IEEE International Conference on Robotics and Automation*, vol. 2, pp. 1018–1023
35. Van den Berg JP, Overmars MH. 2005. Using workspace information as a guide to non-uniform sampling in probabilistic roadmap planners. *International Journal of Robotics Research* 24(12):1055–1071
36. Yang Y, Brock O. 2005. *Effcient Motion Planning Based on Disassembly*. In *Robotics: Science and Systems*. Cambridge, USA

37. Kurniawati H, Hsu D. 2004. *Workspace importance sampling for probabilistic roadmap planning*. In *IEEE International Conference on Intelligent Robots and Systems*, vol. 2, pp. 1618–1623
38. Wilmarth SA, Amato NM, Stiller PF. 1999. *MAPRM: A probabilistic roadmap planner with sampling on the medial axis of the free space*. In *IEEE International Conference on Robotics and Automation*, vol. 2, pp. 1024–1031
39. Burns B, Brock O. 2005. *Toward Optimal Configuration Space Sampling*. In *Robotics: Science and Systems*, pp. 105–112. Cambridge, USA
40. Yershova A, Jaillet L, Siméon T, LaValle SM. 2005. *Dynamic-domain RRTs: Efficient exploration by controlling the sampling domain*. In *IEEE International Conference on Robotics and Automation*, pp. 3856–3861
41. Yershova A, LaValle SM. 2009. Motion planning for highly constrained spaces. In *Robot Motion and Control 2009*, ed. KR Kozłowski, pp. 297–306. London: Springer London
42. Geraerts R, Overmars MH. 2007. Creating high-quality paths for motion planning. *International Journal of Robotics Research* 26(8):845–863
43. Van den Berg J, Overmars M. 2008. Planning time-minimal safe paths amidst unpredictably moving obstacles. *International Journal of Robotics Research* 27(11-12):1274–1294
44. Raveh B, Enosh A, Halperin D. 2011. A little more, a lot better: Improving path quality by a path-merging algorithm. *IEEE Transactions on Robotics* 27(2):365–371
45. Luna R, Şucan IA, Moll M, Kavraki LE. 2013. *Anytime solution optimization for sampling-based motion planning*. In *IEEE International Conference on Robotics and Automation*, pp. 5068–5074
46. Solovey K, Kleinbort M. 2020. The critical radius in sampling-based motion planning. *International Journal of Robotics Research* 39(2-3):266–285
47. Berenson D, Abbeel P, Goldberg K. 2012. *A robot path planning framework that learns from experience*. In *IEEE International Conference on Robotics and Automation*, pp. 3671–3678
48. Coleman D, Sucan IA, Moll M, Okada K, Correll N. 2015. *Experience-based planning with sparse roadmap spanners*. In *IEEE International Conference on Robotics and Automation*, pp. 900–905
49. Lien JM, Lu Y. 2009. Planning motion in environments with similar obstacles. *Robotics: Science and Systems*
50. Lehner P, Albu-Schaeffer A. 2018. The repetition roadmap for repetitive constrained motion planning. *IEEE Robotics and Automation Letters* 3(3):3884–3891
51. Chamzas C, Kingston Z, Quintero-Peña C, Shrivastava A, E. Kavraki L. 2021. *Learning Sampling Distributions Using Local 3D Workspace Decompositions for Motion Planning in High Dimensions*. In *IEEE International Conference on Robotics and Automation*, pp. 1283–1289
52. Ichter B, Harrison J, Pavone M. 2018. *Learning sampling distributions for robot motion planning*. In *IEEE International Conference on Robotics and Automation*, pp. 7087–7094
53. Chamzas C, Cullen A, Shrivastava A, E. Kavraki L. 2022. *Learning to Retrieve Relevant Experiences for Motion Planning*. In *IEEE International Conference on Robotics and Automation*, pp. 7233–7240
54. McMahon T, Sivaramakrishnan A, Granados E, Bekris KE. 2022. A survey on the integration of machine learning with sampling-based motion planning. *Foundations and Trends in Robotics* 9(4):266–327
55. Choset H, Lynch KM, Hutchinson S, Kantor GA, Burgard W. 2005. *Principles of robot motion: theory, algorithms, and implementations*. MIT press
56. LaValle SM. 2006. *Planning Algorithms*. Cambridge University Press
57. Lynch KM, Park FC. 2017. *Modern robotics*. Cambridge University Press
58. Farber M. 2003. Topological complexity of motion planning. *Discrete and Computational Geometry* 29(2):211–221
59. Lee JM. 2003. *Introduction to Smooth Manifolds*. New York, NY: Springer New York

60. Čech E, Frolík Z, Katětov M. 1966. *Topological spaces*. Academia, Publishing House of the Czechoslovak Academy of Sciences
61. Wilson WA. 1931. On quasi-metric spaces. *American Journal of Mathematics* 53(3):675–684
62. Grothe F, Hartmann VN, Orthey A, Toussaint M. 2022. *ST-RRT*: Asymptotically-Optimal Bidirectional Motion Planning through Space-Time*. In *IEEE International Conference on Robotics and Automation*, pp. 3314–3320
63. Wilson WA. 1931. On semi-metric spaces. *American Journal of Mathematics* 53(2):361–373
64. Wang G, Zhang B, Ng TE. 2007. *Towards network triangle inequality violation aware distributed systems*. In *ACM SIGCOMM Conference on Internet Measurement*, pp. 175–188
65. Ladd AM, Kavraki LE. 2004. Fast tree-based exploration of state space for robots with dynamics. In *Algorithmic Foundations of Robotics VI*, ed. M Erdmann, M Overmars, D Hsu, F van der Stappen, pp. 297–312. Springer
66. Bertsekas D, Tsitsiklis JN. 2008. *Introduction to probability*, vol. 1. Athena Scientific
67. Verginis CK, Dimarogonas DV, Kavraki LE. 2023. KDF: Kinodynamic Motion Planning via Geometric Sampling-Based Algorithms and Funnel Control. *IEEE Transactions on Robotics* 39(2):978–997
68. Janson L, Ichter B, Pavone M. 2018. Deterministic sampling-based motion planning: Optimality, complexity, and performance. *The International Journal of Robotics Research* 37(1):46–61
69. Palmieri L, Bruns L, Meurer M, Arras KO. 2019. Dispertio: Optimal sampling for safe deterministic motion planning. *IEEE Robotics and Automation Letters* 5(2):362–368
70. Choudhury S, Gammell JD, Barfoot TD, Srinivasa SS, Scherer S. 2016. *Regionally accelerated batch informed trees (RABIT): A framework to integrate local information into optimal path planning*. In *IEEE International Conference on Robotics and Automation*, pp. 4207–4214
71. Faust A, Ramirez O, Fiser M, Oslund K, Francis A, et al. 2018. *PRM-RL: Long-range Robotic Navigation Tasks by Combining Reinforcement Learning and Sampling-based Planning*. In *IEEE International Conference on Robotics and Automation*, pp. 5113–5120
72. Dobson A, Bekris KE. 2014. Sparse roadmap spanners for asymptotically near-optimal motion planning. *International Journal of Robotics Research* 33(1):18–47
73. Salzman O, Hemmer M, Halperin D. 2013. On the power of manifold samples in exploring configuration spaces and the dimensionality of narrow passages. In *Algorithmic Foundations of Robotics X*, ed. E Frazzoli, T Lozano-Perez, N Roy, D Rus, pp. 313–329. Berlin, Heidelberg: Springer Berlin Heidelberg
74. Janson L, Schmerling E, Clark A, Pavone M. 2015. Fast marching tree: A fast marching sampling-based method for optimal motion planning in many dimensions. *International Journal of Robotics Research* 34(7):883–921
75. Bohlin R, Kavraki LE. 2000. *Path planning using lazy PRM*. In *IEEE International Conference on Robotics and Automation*, vol. 1, pp. 521–528
76. Hauser K. 2015. *Lazy collision checking in asymptotically-optimal motion planning*. In *IEEE International Conference on Robotics and Automation*, pp. 2951–2957
77. Mandalika A, Choudhury S, Salzman O, Srinivasa S. 2019. *Generalized lazy search for robot motion planning: Interleaving search and edge evaluation via event-based toggles*. In *International Conference on Automated Planning and Scheduling*, vol. 29, pp. 745–753
78. Orthey A, Toussaint M. 2019. Rapidly-exploring quotient-space trees: Motion planning using sequential simplifications. In *ISRR 2019: Robotics Research*, ed. T Asfour, E Yoshida, J Park, H Christensen, O Khatib, pp. 52–68. Springer International Publishing
79. Klemm S, Oberländer J, Hermann A, Roennau A, Schamm T, et al. 2015. *RRT*-Connect: Faster, asymptotically optimal motion planning*. In *IEEE International Conference on Robotics and Biomimetics (ROBIO)*, pp. 1670–1677
80. Siméon T, Laumond JP, Nissoux C. 2000. Visibility-based probabilistic roadmaps for motion planning. *Advanced Robotics* 14(6):477–493
81. Li Y, Littlefield Z, Bekris KE. 2016. Asymptotically optimal sampling-based kinodynamic

planning. *International Journal of Robotics Research* 35(5):528–564
82. Salzman O, Halperin D. 2016. Asymptotically near-optimal RRT for fast, high-quality motion planning. *IEEE Transactions on Robotics* 32(3):473–483
83. Pearl J. 1984. *Heuristics: intelligent search strategies for computer problem solving*. Addison-Wesley Longman Publishing Co., Inc.
84. Gammell JD, Srinivasa SS, Barfoot TD. 2014. *Informed RRT*: Optimal sampling-based path planning focused via direct sampling of an admissible ellipsoidal heuristic*. In *IEEE International Conference on Intelligent Robots and Systems*, pp. 2997–3004
85. Gammell JD, Barfoot TD, Srinivasa SS. 2020. Batch informed trees (BIT*): Informed asymptotically optimal anytime search. *International Journal of Robotics Research* 39(5):543–567
86. Strub MP, Gammell JD. 2020. *Advanced BIT*(ABIT*): Sampling-Based Planning with Advanced Graph-Search Techniques*. In *IEEE International Conference on Robotics and Automation*, pp. 130–136
87. Şucan IA, Kavraki LE. 2009. Kinodynamic motion planning by interior-exterior cell exploration. In *Algorithmic Foundation of Robotics VIII*, ed. GS Chirikjian, H Choset, M Morales, T Murphey, pp. 449–464. Springer
88. Şucan IA, Kavraki LE. 2011. A sampling-based tree planner for systems with complex dynamics. *IEEE Transactions on Robotics* 28(1):116–131
89. Reid W, Fitch R, Göktoğan AH, Sukkarieh S. 2019. Sampling-based hierarchical motion planning for a reconfigurable wheel-on-leg planetary analogue exploration rover. *Journal of Field Robotics* 37(5):786–811
90. Şucan IA, Moll M, Kavraki L. 2012. The open motion planning library. *IEEE Robotics and Automation Magazine* 19(4):72–82
91. Moll M, Chamzas C, Kingston Z, Kavraki LE. 2021. *HyperPlan: A Framework for Motion Planning Algorithm Selection and Parameter Optimization*. In *IEEE International Conference on Intelligent Robots and Systems*, pp. 2511–2518
92. Cano J, Yang Y, Bodin B, Nagarajan V, O'Boyle M. 2018. *Automatic parameter tuning of motion planning algorithms*. In *IEEE International Conference on Intelligent Robots and Systems*, pp. 8103–8109
93. Gammell JD, Strub MP. 2021. A survey of asymptotically optimal sampling-based motion planning methods. *Annual Review of Control, Robotics, and Autonomous Systems* 4(1):295–318
94. Gammell JD, Barfoot TD, Srinivasa SS. 2018. Informed sampling for asymptotically optimal path planning. *IEEE Transactions on Robotics* 34(4):966–984
95. Orthey A, Toussaint M. 2021. *Sparse Multilevel Roadmaps for High-Dimensional Robot Motion Planning*. In *IEEE International Conference on Robotics and Automation*, pp. 7851–7857
96. McCarthy Z, Bretl T, Hutchinson S. 2012. *Proving path non-existence using sampling and alpha shapes*. In *IEEE International Conference on Robotics and Automation*, pp. 2563–2569
97. Varava A, Carvalho JF, Pokorny FT, Kragic D. 2021. Free space of rigid objects: Caging, path non-existence, and narrow passage detection. *International Journal of Robotics Research* 40(10-11):1049–1067
98. Li S, Dantam N. 2021. *Learning Proofs of Motion Planning Infeasibility*. In *Robotics: Science and Systems*
99. Dubins LE. 1957. On curves of minimal length with a constraint on average curvature, and with prescribed initial and terminal positions and tangents. *American Journal of mathematics* 79(3):497–516
100. Reeds J, Shepp L. 1990. Optimal paths for a car that goes both forwards and backwards. *Pacific Journal of Mathematics* 145(2):367–393
101. Hauser K, Zhou Y. 2016. Asymptotically optimal planning by feasible kinodynamic planning in a state–cost space. *IEEE Transactions on Robotics* 32(6):1431–1443
102. Mukadam M, Dong J, Yan X, Dellaert F, Boots B. 2018. Continuous-time gaussian pro-

cess motion planning via probabilistic inference. *International Journal of Robotics Research* 37(11):1319–1340

103. Kamat J, Ortiz-Haro J, Toussaint M, Pokorny FT, Orthey A. 2022. *BITKOMO: Combining Sampling and Optimization for Fast Convergence in Optimal Motion Planning*. In *IEEE International Conference on Intelligent Robots and Systems*, pp. 4492–4497
104. Alwala KV, Mukadam M. 2021. *Joint sampling and trajectory optimization over graphs for online motion planning*. In *IEEE International Conference on Intelligent Robots and Systems*, pp. 4700–4707
105. Kingston Z, Moll M, Kavraki LE. 2018. Sampling-based methods for motion planning with constraints. *Annual Review of Control, Robotics, and Autonomous Systems* 1(1):159–185
106. Berenson D, Srinivasa S, Kuffner J. 2011. Task space regions: A framework for pose-constrained manipulation planning. *International Journal of Robotics Research* 30(12):1435–1460
107. Phillips M, Likhachev M. 2011. *Sipp: Safe interval path planning for dynamic environments*. In *IEEE International Conference on Robotics and Automation*, pp. 5628–5635
108. Otte M, Frazzoli E. 2016. RRTX: Asymptotically optimal single-query sampling-based motion planning with quick replanning. *International Journal of Robotics Research* 35(7):797–822
109. Kurniawati H. 2022. Partially observable markov decision processes and robotics. *Annual Review of Control, Robotics, and Autonomous Systems* 5:253–277
110. Phiquepal C, Orthey A, Viennot N, Toussaint M. 2022. Path-tree optimization in discrete partially observable environments using rapidly-exploring belief-space graphs. *IEEE Robotics and Automation Letters* 7(4):10160–10167
111. Elimelech K, Indelman V. 2022. Simplified decision making in the belief space using belief sparsification. *International Journal of Robotics Research* 41(5):470–496
112. Ko I, Kim B, Park FC. 2014. Randomized path planning on vector fields. *International Journal of Robotics Research* 33(13):1664–1682
113. Tang Z, Chen B, Lan R, Li S. 2020. Vector field guided RRT* based on motion primitives for quadrotor kinodynamic planning. *Journal of Intelligent & Robotic Systems* 100:1325–1339
114. Hernández JD, Vidal E, Moll M, Palomeras N, Carreras M, Kavraki LE. 2019. Online motion planning for unexplored underwater environments using autonomous underwater vehicles. *Journal of Field Robotics* 36(2):370–396
115. Vidal E, Moll M, Palomeras N, Hernández JD, Carreras M, Kavraki LE. 2019. *Online multilayered motion planning with dynamic constraints for autonomous underwater vehicles*. In *IEEE International Conference on Robotics and Automation*, pp. 8936–8942
116. Boyd S, Vandenberghe L. 2004. *Convex Optimization*. Cambridge University Press
117. Hauser K. 2014. Fast interpolation and time-optimization with contact. *International Journal of Robotics Research* 33(9):1231–1250
118. Zucker M, Ratliff N, Dragan AD, Pivtoraiko M, Klingensmith M, et al. 2013. CHOMP: Covariant Hamiltonian Optimization for Motion Planning. *International Journal of Robotics Research* 32(9-10):1164–1193
119. Schulman J, Duan Y, Ho J, Lee A, Awwal I, et al. 2014. Motion planning with sequential convex optimization and convex collision checking. *International Journal of Robotics Research* 33(9):1251–1270
120. Schaal S. 2006. Dynamic movement primitives-a framework for motor control in humans and humanoid robotics. In *Adaptive motion of animals and machines*. Springer
121. Kober J, Peters J. 2011. Policy search for motor primitives in robotics. *Machine Learning* 84(1-2):171–203
122. Kober J, Bagnell JA, Peters J. 2013. Reinforcement learning in robotics: A survey. *International Journal of Robotics Research* 32(11):1238–1274
123. Paraschos A, Daniel C, Peters J, Neumann G. 2018. Using probabilistic movement primitives in robotics. *Autonomous Robots* 42:529–551

124. Giszter SF. 2015. Motor primitives—new data and future questions. *Current Opinion in Neurobiology* 33:156–165
125. Graziano M. 2008. *The intelligent movement machine: An ethological perspective on the primate motor system*. Oxford University Press
126. Bhardwaj M, Sundaralingam B, Mousavian A, Ratliff ND, Fox D, et al. 2022. *Storm: An integrated framework for fast joint-space model-predictive control for reactive manipulation*. In *Conference on Robot Learning*, pp. 750–759. PMLR
127. Koenig S, Likhachev M, Furcy D. 2004. Lifelong planning A*. *Artificial Intelligence* 155(1-2):93–146
128. Ren Z, Rathinam S, Likhachev M, Choset H. 2022. Multi-objective path-based D* lite. *IEEE Robotics and Automation Letters* 7(2):3318–3325
129. Hart PE, Nilsson NJ, Raphael B. 1968. A formal basis for the heuristic determination of minimum cost paths. *IEEE transactions on Systems Science and Cybernetics* 4(2):100–107
130. Liu S, Mohta K, Atanasov N, Kumar V. 2018. Search-based motion planning for aggressive flight in SE(3). *IEEE Robotics and Automation Letters* 3(3):2439–2446
131. Araki M. 2010. Control systems, robotics and automation—vol ii—PID control. *Kyoto University, Japan*
132. Klemm V, Morra A, Gulich L, Mannhart D, Rohr D, et al. 2020. LQR-assisted whole-body control of a wheeled bipedal robot with kinematic loops. *IEEE Robotics and Automation Letters* 5(2):3745–3752
133. Grandia R, Farshidian F, Ranftl R, Hutter M. 2019. *Feedback MPC for torque-controlled legged robots*. In *IEEE International Conference on Intelligent Robots and Systems*, pp. 4730–4737
134. Majumdar A, Tedrake R. 2017. Funnel libraries for real-time robust feedback motion planning. *International Journal of Robotics Research* 36(8):947–982
135. Sutton RS, Barto AG. 2018. *Reinforcement learning: An introduction*. MIT press
136. Watkins CJCH. 1989. Learning from delayed rewards. Ph.D. thesis, King's College, Cambridge United Kingdom
137. Moll M, Şucan IA, Kavraki LE. 2015. Benchmarking motion planning algorithms: An extensible infrastructure for analysis and visualization. *IEEE Robotics and Automation Magazine* 22(3):96–102
138. Hsu D, Latombe JC, Kurniawati H. 2006. On the probabilistic foundations of probabilistic roadmap planning. *International Journal of Robotics Research* 25(7):627–643
139. Kingston Z, Moll M, Kavraki LE. 2019. Exploring implicit spaces for constrained sampling-based planning. *International Journal of Robotics Research* 38(10–11):1151–1178
140. Tsianos KI, Sucan IA, Kavraki LE. 2007. Sampling-based robot motion planning: Towards realistic applications. *Computer Science Review* 1(1):2–11
141. Elbanhawi M, Simic M. 2014. Sampling-based robot motion planning: A review. *IEEE Access* 2:56–77
142. Mac TT, Copot C, Tran DT, De Keyser R. 2016. Heuristic approaches in robot path planning: A survey. *Robotics and Autonomous Systems* 86:13–28
143. 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
144. Al-Bluwi I, Siméon T, Cortés J. 2012. Motion planning algorithms for molecular simulations: A survey. *Computer Science Review* 6(4):125–143
145. Goerzen C, Kong Z, Mettler B. 2010. A survey of motion planning algorithms from the perspective of autonomous uav guidance. *Journal of Intelligent and Robotic Systems* 57(1):65–100
146. Garrett CR, Chitnis R, Holladay R, Kim B, Silver T, et al. 2021. Integrated task and motion planning. *Annual Review of Control, Robotics, and Autonomous Systems* 4:265–293