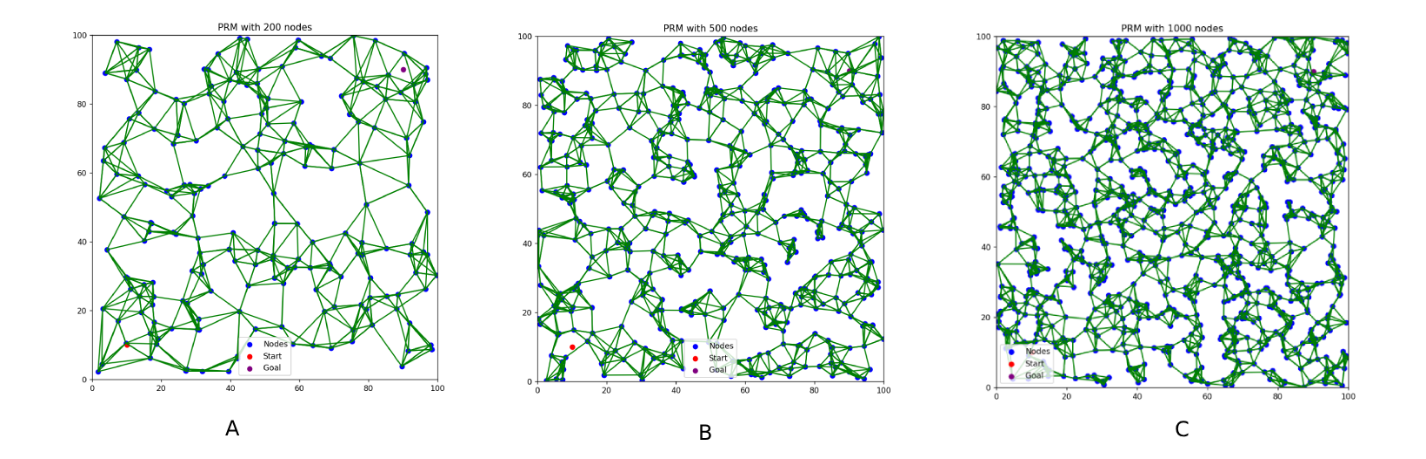
# Literature Review

### 2. Literature Review

#### 2.1 Integrated Grasp and Motion Planning: Setting the Context

The field of robotics has seen significant advancements in recent years, particularly in the domain of integrated grasp and motion planning. While motion planning is a fundamental aspect of robotics (Elbanhawi & Simić, 2014), the simultaneous planning of both grasping an object and coordinating the robot's motion remains a notable challenge (Bütepage et al., 2019). This challenge is compounded by computational obstacles related to sensing, grasp analysis, motion planning, and the execution of the robot arm's movements (Ichnowski et al., 2020).

Although grasping and motion planning have traditionally been treated as separate problems, their integration is essential for real-world robotic applications across industries such as manufacturing and autonomous systems. The exploration of algorithms for solving this dual problem began with a focus on probabilistic approaches. Probabilistic Roadmap Methods (PRM) were considered initially due to their efficiency in navigating high-dimensional spaces by constructing roadmaps from random samples (Kavraki et al., 1996). However, PRMs are often less effective in dynamic environments, where grasp conditions and motion tasks change rapidly. These limitations highlighted the need for an algorithm capable of more flexible, real-time planning.



This led to a deeper investigation of the Rapidly Exploring Random Tree (RRT) algorithm, known for its ability to efficiently explore high-dimensional spaces without requiring pre-constructed paths. RRT incrementally builds a tree, exploring the space as needed, making it ideal for dynamic tasks requiring responsive, real-time path planning (LaValle, 1998). Its adaptability, in contrast to PRM, made it a strong candidate for integrated grasp and motion planning.

Further investigation into RRT's extensions revealed promising advancements. Algorithms like RRT\*, JPplusRRT, IK-RRT, and BIK-RRT introduced improvements such as optimality and enhanced adaptability. These algorithms build on the core principles of RRT, refining its efficiency and effectiveness in addressing the combined challenges of grasp and motion planning. For example, RRT\* offers optimal paths, while IK-RRT focuses on handling complex inverse kinematics for grasping.

The evolution from probabilistic roadmaps to dynamic algorithms like RRT and its extensions reflects the ongoing progress in the field of robotics. This literature review sets the stage by exploring these advancements and highlighting the potential of these algorithms to solve integrated grasp and motion planning challenges, forming the basis for the research questions and methodology.

#### 2.2 Algorithms

In this section, several sampling-based motion planning algorithms are discussed, beginning with foundational methods and leading into more advanced variations that address specific limitations. First, Configuration space and the essential primitive procedures that underlie these algorithms are introduced. Then, two of the major paradigms for sampling-based motion planning are presented: Probabilistic Roadmaps (PRM) and Rapidly Exploring Random Trees (RRT). Finally, more advanced algorithms, namely RRT\*, JPlusRRT, IK-RRT and BIK-RRT, are introduced, which provide asymptotic optimality and improved computational efficiency over their "standard" counterparts.

#### 2.2.1 Configuration Space

A key concept in motion planning is the **configuration space (C-space)**. The configuration space represents all possible positions and orientations a robot can occupy. A specific configuration is defined by a set of variables, such as the robot's joint angles or positions in the workspace. The free configuration space consists of all configurations where the robot does not collide with obstacles, while the obstacle space represents configurations where collisions occur.

Motion planning algorithms, including the ones discussed in this section, operate within searching for a collision-free path from a start configuration to a goal configuration . The complexity of navigating ​, especially in high-dimensional spaces, forms the basis for developing efficient algorithms like PRM and RRT, which use random sampling to explore feasible paths.

#### 2.2.2 Primitive Procedures

Before discussing the algorithms themselves, it is useful to define several primitive operations that all these sampling-based algorithms rely upon. These include sampling, nearest neighbor searches, and collision detection:

**Sampling**: A random sampling of the configuration space, denoted as , is the core of all sampling-based methods. The procedure, denoted as , generates independent and identically distributed (i.i.d.) points from . An extension of this, , ensures that samples are drawn from the free space ​, avoiding obstacles.

**Nearest Neighbor**: Given a graph where , the nearest neighbor function, , returns the vertex that is closest to the point , based on a distance metric, often Euclidean.

**Steering**: This function, , generates a point that moves closer to the target from the point , while maintaining a maximum allowable step size, ensuring gradual and feasible transitions.

**Collision Test**: The function evaluates whether the straight-line path between two points lies entirely within ​.

These primitive operations serve as the building blocks for the sampling-based motion planning algorithms discussed below (Karaman and Frazzoli, 2011).

#### 2.2.3 Probabilistic Roadmaps (PRM)

The Probabilistic Roadmap (PRM) algorithm, introduced by Kavraki et al. (1996), is designed primarily for multi-query applications. It begins with a pre-processing phase in which a roadmap is constructed by randomly sampling points in the free configuration space, ​, and attempting to connect them using a local planner (e.g., straight-line connections). The graph, or roadmap, built in this phase is a collection of nodes connected by edges, representing collision-free paths between the nodes. Once the roadmap is built, PRM can be used to solve multiple path queries efficiently by simply searching for a path through the pre-constructed roadmap.

PRM is effective for solving motion planning problems in static environments, but its reliance on pre-processing makes it less suited for real-time applications where the environment may change. The method focuses primarily on establishing connectivity between different regions of the configuration space, making it well-suited for environments where multiple paths are frequently queried. The pre-processing phase of the PRM algorithm is outlined in the following pseudocode:

|  |
| --- |
|  |
| 1  2  3  4  5  6  7  8  9 |

This pseudocode describes the pre-processing phase of the PRM algorithm, where nodes are sampled, checked for connectivity, and added to the roadmap if they meet the collision-free criteria. After constructing the roadmap, it is ready for efficient querying in the second phase of the PRM algorithm.

#### 2.2.4 Overview of the RRT Algorithm

The Rapidly Exploring Random Tree (RRT) algorithm, originally proposed by LaValle (1998), was developed to solve motion planning problems in high-dimensional spaces. Unlike traditional methods that rely on pre-constructed paths or roadmaps, RRT incrementally builds a tree structure by randomly sampling points from the free configuration space, , and connecting each sample to the nearest node already in the tree. This incremental process enables the algorithm to quickly explore vast, complex environments without prior knowledge of the entire space.

RRT is mainly suited for single-query applications, where the objective is to find a feasible path from the start to the goal configuration. The algorithm starts by initializing a graph with the initial configuration as the sole vertex, and no edges. At each iteration, a random point is sampled. The nearest node v ∈ V from the current tree is then identified, and an attempt is made to steer from v toward x\_rand. If this new connection is valid (i.e., it avoids obstacles), the new point x\_new is added to the tree as a vertex, and the edge (v, x\_new) is added to the edge set.

This process continues until a specified number of iterations n is completed, or the tree reaches the goal region. In the original RRT algorithm, the iteration could stop as soon as a path to the goal is found. However, for consistency with other algorithms such as PRM, the iteration is typically performed for a set number of steps. In the absence of obstacles (i.e., when X\_free = X), the constructed tree essentially forms an online nearest-neighbor graph.

While RRT efficiently finds feasible paths, it does not ensure that the path is optimal in terms of distance or smoothness. The random nature of the algorithm enables rapid exploration, but can also lead to suboptimal solutions, with paths that may be unnecessarily long or inefficient. This limitation has spurred the development of variants like RRT\*, which aims to improve path quality by optimizing the connections between nodes.

The pseudocode for the basic RRT algorithm is as follows:

|  |
| --- |
| Algorithm 2: RRT |
|  |

#### 2.2.5 RRT\*: Optimal Path Planning

To address the lack of optimality in the original RRT, RRT\* was developed by Karaman and Frazzoli (2011). This extension of RRT improves the algorithm by adding a mechanism for finding not just any valid path, but the most efficient one in terms of distance or cost. RRT\* continually refines the path as the tree expands, reconnecting nodes and edges to ensure that the path approaches optimality over time. However, this refinement process comes at a cost—while RRT\* guarantees an optimal path, it often takes significantly more time to do so compared to the original RRT. The trade-off between computational time and path optimality is an unresolved tension within RRT\*’s application to real-world tasks, particularly when real-time decisions are critical.

#### 2.2.6 JPplusRRT: Enhanced Speed and Adaptability

JPplusRRT is an extension that attempts to balance the exploration efficiency of RRT with the need for faster decision-making in dynamic environments. While RRT\* is more focused on finding optimal paths, JPplusRRT sacrifices some of that optimality to speed up exploration and adaptability. In tasks that involve both grasping and motion, where real-time adaptability to new conditions is crucial, JPplusRRT offers a more pragmatic solution. Its speed allows it to quickly adapt to changing environments, updating the path without being bogged down by the pursuit of an ideal path. Yet, the question remains: is the sacrifice of path quality justified by the increased speed? This tension lingers, as the algorithm navigates the space between efficiency and precision.

#### 2.2.7 IK-RRT and BIK-RRT: Addressing Complex Configurations

When integrated with grasp planning, handling the complexities of inverse kinematics (IK) becomes crucial. IK-RRT and BIK-RRT are two algorithms that extend RRT by incorporating solutions for complex configuration spaces, specifically those that arise in tasks requiring precise grasping. IK-RRT introduces a mechanism to solve inverse kinematics problems on the fly, allowing the robot to not only find a path to the object but to do so while also determining how to grasp it. BIK-RRT, on the other hand, adds a bidirectional approach to further improve efficiency, growing trees from both the start and goal configurations, meeting in the middle.

Yet, as these algorithms become more specialized, they also become more cumbersome. The complexity of solving inverse kinematics while simultaneously exploring the configuration space often slows down the entire process. While IK-RRT and BIK-RRT provide solutions to complex manipulation tasks, their utility in time-sensitive environments remains a point of contention.

#### 2.3 Comparative Analysis of Motion Planning Algorithms

* 2.3.1 Criteria for Comparison: Planning Time, Success Rate, and Exploration
* 2.3.2 Testing in Shared Environments: Consistency and Parameters
* 2.3.3 Performance Insights from the Literature

#### 2.4 Grasp Selection in Motion Planning

* 2.4.1 What is a Grasp? Definition and Importance
* 2.4.2 Grasp Planner: Selection Process and Technical Steps

#### 2.5 Other Relevant Algorithms in Grasp and Motion Planning

## 2.7 Gaps in literature

## 2.8 summary