**A logo for a university

Description automatically generated**

*Integrated grasp and motion planning*

**Author:**

Salah Ebrahimpour

**Programme:**

MSc Artificial Intelligence with Business Strategy

**Supervisor:**

Dr. Martin Rudorfer

**Co-Supervisor:**

Dr. Anthony Henry

**Date:**

September 2024

**Abstract**

**Acknowledgement**

I would like to express my deepest gratitude to my supervisors, Dr. Martin Rudorfer and Dr. Anthony Henry, for their invaluable guidance, support, and expertise throughout this project. Their insights and encouragement were instrumental in shaping the direction of this dissertation, and I am incredibly grateful for their patience and mentorship.

I would also like to thank my family and friends for their constant support and encouragement, providing me with the motivation to keep going, especially during challenging times. Special thanks to my colleagues and peers at Aston University for their collaboration and feedback, which contributed significantly to the development of my research.

Finally, I extend my appreciation to the faculty and staff of the MSc Artificial Intelligence with Business Strategy program for providing a conducive environment for learning and research. This journey has been an enriching experience, and I am thankful to everyone who has played a part in it.

Contents

[Chapter 1: Introduction 1](#_Toc178328299)

[1.1. Project background 1](#_Toc178328300)

[1.2. Research Objectives 1](#_Toc178328301)

[1.3. Research Questions/Hypotheses 2](#_Toc178328302)

[1.4. Significance of the Study 3](#_Toc178328303)

[1.5. Scope and Limitations 4](#_Toc178328304)

[1.6. Thesis Structure 4](#_Toc178328305)

[Chapter 2: Literature Review 5](#_Toc178328306)

[2.1 Integrated Grasp and Motion Planning: Setting the Context 5](#_Toc178328307)

[2.2 Algorithms 7](#_Toc178328308)

[2.2.1 Configuration Space 7](#_Toc178328309)

[2.2.2 Primitive Operations 7](#_Toc178328310)

[2.2.3 Probabilistic Roadmaps (PRM) 8](#_Toc178328311)

[2.2.4 Overview of the RRT Algorithm 8](#_Toc178328312)

[2.2.5 RRT\*: Optimal Path Planning 9](#_Toc178328313)

[2.2.6 JPplusRRT: Enhanced Speed and Adaptability 12](#_Toc178328314)

[2.2.7 IK-RRT and BIK-RRT: Addressing Complex Configurations 14](#_Toc178328315)

[2.3 Comparative Analysis of Motion Planning Algorithms 14](#_Toc178328316)

[2.3.1 Criteria for Comparison: Planning Time, Success Rate, and Exploration 14](#_Toc178328317)

[2.3.2 Testing in Shared Environments: Consistency and Parameters 15](#_Toc178328318)

[2.3.3 Performance Insights 16](#_Toc178328319)

[2.4 Grasp Selection in Motion Planning 17](#_Toc178328320)

[2.4.1 What is a Grasp? Definition and Importance 17](#_Toc178328321)

[2.4.2 Grasp Planner: Selection Process and Technical Steps 17](#_Toc178328322)

[2.5 Other Algorithms in Grasp and Motion Planning 17](#_Toc178328323)

[2.6 Gaps in literature 17](#_Toc178328324)

[2.7 summary 17](#_Toc178328325)

[Chapter 3: Problem Description 17](#_Toc178328326)

[Chapter 4: Methodology 19](#_Toc178328327)

[4.1 Introduction 19](#_Toc178328328)

[4.2 Problem Formulation 20](#_Toc178328329)

[4.3 Research Design 21](#_Toc178328330)

[4.4 Algorithms and Frameworks 22](#_Toc178328331)

[4.5 Experimental Setup 25](#_Toc178328332)

[4.6. Metrics for Evaluation 26](#_Toc178328333)

[4.9. Data Collection 27](#_Toc178328334)

[4.9. Challenges and Adaptations 27](#_Toc178328335)

[Chapter 5: Project Management 28](#_Toc178328336)

[Chapter 6: Business Strategy and Market Integration 28](#_Toc178328337)

[Introduction to Business Strategy 28](#_Toc178328338)

[Robotics Marketplace and Its Growth Potential 28](#_Toc178328339)

[Market Landscape and Competitive Analysis 28](#_Toc178328340)

[Competitive Advantage 28](#_Toc178328341)

[Commercialization Strategy 28](#_Toc178328342)

[Industry Partnerships and Collaboration 28](#_Toc178328343)

[Chapter 8: Conclusion 29](#_Toc178328344)

[References 30](#_Toc178328345)

[Appendix 40](#_Toc178328346)

**Figures**

[Figure 1 Gantt Chart for Dissertation on Integrated Grasp and Motion Planning. 27](#_Toc177638490)

[Figure 2 Porter's Five forces analysis of Integrated grasp and motion planning 35](#_Toc177638491)

**Abbreviations**

# Chapter 1: Introduction

## Project background

As robotics continues to evolve, the need for more sophisticated systems capable of autonomous, complex manipulation tasks becomes ever more pressing. Traditionally, the processes of grasp and motion planning have been treated separately in robotic systems. Grasp planning determines how a robot should grip an object based on factors like the object's shape, weight, and texture. Motion planning, on the other hand, focuses on the robot's path and the obstacles it must avoid when executing tasks (Muhayyuddin et al., 2015). However, the separation of these two processes can lead to suboptimal performance, particularly in dynamic environments where grasp configuration can affect motion feasibility and vice versa (Ali & Lee, 2020).

A robotic arm in a warehouse

Description automatically generated

To address these issues, researchers have started exploring integrated approaches, where grasp and motion planning are considered simultaneously. These unified systems aim to enhance efficiency, adaptability, and reliability in robotic tasks. For example, the Grasp-RRT approach represents one such integration, optimizing both grasp and motion planning within the same framework (Ali & Lee, 2020).

## Research Objectives

The main objective of this research is to implement several motion-planning algorithms—specifically RRT, RRT\*, JPplusRRT, IK-RRT, and BIK-RRT—within the context of integrated grasp and motion planning tasks. This study aims to assess these algorithms based on key performance metrics such as planning time, success rate, and path exploration efficiency. By implementing these algorithms across multiple predefined scenarios, the research seeks to understand the complexity of their implementation and the trade-offs they present between computational efficiency and the quality of the generated paths. Additionally, this comparison will provide insights into each algorithm's adaptability to different environmental configurations, including variations in object placement and obstacle density.

In a further phase of the research, the RRT\* algorithm is implemented in a second, more complex environment provided by the Jogramop Framework (Rudorfer et al., 2024), which offers 20 standardized benchmark scenarios. This implementation will allow for a detailed comparison of RRT\*’s performance against other algorithms, such as JPplusRRT and IK-RRT, under consistent benchmark conditions. The research ultimately aims to provide insights into the strengths and limitations of RRT\* in both standard and complex environments, contributing to the development of more efficient and robust integrated grasp and motion planning strategies.

## Research Questions/Hypotheses

* How do the RRT, RRT\*, JPplusRRT, IK-RRT, and BIK-RRT algorithms compare in terms of planning time and success rate across different scenarios?
* What specific modifications are required to transform the base RRT algorithm into RRT\*, JPplusRRT, IK-RRT, and BIK-RRT, and how do these changes impact performance?
* What insights can be gained by analysing the path quality and search space exploration characteristics of RRT and these other algorithms across different benchmark scenarios?

This research focuses on understanding the core principles of the RRT algorithm and how it has been modified to create more advanced algorithms such as RRT\*, JPplusRRT, IK-RRT, and BIK-RRT. The first research question seeks to compare these algorithms based on key metrics like planning time and success rate in different environments. Since these algorithms are built upon the base RRT, the second question investigates the specific changes—such as improving optimality in RRT\* or handling more complex configurations in JPplusRRT—and how these modifications affect performance and outcomes.

Further, the third research question explores the environmental adaptability of each algorithm, analysing which perform best in dynamic, cluttered, or constrained spaces. This question aims to determine where each algorithm excels or faces challenges, offering practical insights into their application in real-world scenarios. Lastly, the fourth question examines the path planning quality of each algorithm, such as smoothness and efficiency, and how they explore the search space, providing deeper insights into the trade-offs between path quality and computational performance across benchmark scenarios.

## Significance of the Study

This research focuses on evaluating the performance of several motion-planning algorithms—RRT, RRT\*, JPplusRRT, IK-RRT, and BIK-RRT—particularly in how they handle grasp and motion planning in challenging environments. By assessing key metrics such as planning time, success rate, and adaptability, the study provides critical insights into the strengths and limitations of these algorithms when applied in real-world scenarios requiring efficient and precise motion planning.

A notable gap in the current literature is that many algorithms exist only as pseudocode, without full implementation in simulated or real-world environments. Furthermore, the fine-tuning parameters used to optimize these algorithms are often not provided, leaving uncertainty about their actual performance. This research addresses these issues by implementing these algorithms in simulated environments, such as PyBullet, and benchmarking them using frameworks like Jogramop (Rudorfer et al., 2024). By doing so, the study provides valuable data on how these algorithms perform when implemented and fine-tuned in controlled benchmark scenarios.

The findings have broader implications for improving robotic systems' performance, offering a more reliable understanding of how these algorithms operate in environments with complex constraints. By comparing their effectiveness and providing recommendations for enhancement, this research contributes to the development of more robust and adaptable robotic systems, which can execute real-world tasks with greater efficiency and precision across various industries.

## Scope and Limitations

This research focuses on the development and evaluation of motion-planning algorithms for integrated grasp and motion planning, specifically in the context of robotic manipulation tasks. The algorithms under investigation include RRT, RRT\*, JPplusRRT, IK-RRT, and BIK-RRT. The scope of this study involves implementing and comparing these algorithms in simulated environments, such as PyBullet, and using benchmark scenarios provided by frameworks like Jogramop (Rudorfer et al., 2024). The simulated environments allow for repeatability and precise control over experimental variables, ensuring that each algorithm is tested under consistent conditions to provide meaningful comparisons of performance metrics such as planning time, success rate, and path quality.

However, the reliance on simulation introduces certain limitations. While simulations provide a controlled environment for testing and fine-tuning algorithms, they may not fully capture the complexities of real-world applications. In practical settings, factors such as sensor inaccuracies, mechanical limitations of robots, or unforeseen environmental constraints could impact the performance of the algorithms. Additionally, fine-tuning parameters in simulations may not directly translate to real-world applications, where noise and uncertainty play a greater role. These limitations highlight the need for future research to validate the proposed algorithms in physical environments, addressing the challenges of real-world robotic systems to fully assess their practical viability.

## Thesis Structure

This dissertation is structured across several chapters that comprehensively examine the integration of grasp and motion planning in robotics. The organization is as follows: Chapter 1 introduces the project, outlining the background, research objectives, and key questions guiding the study. Chapter 2 presents a literature review, examining previous work on grasp and motion planning, with an emphasis on integrated approaches and existing gaps. Chapter 3 defines the problem and research gap addressed by the study. Chapter 4 describes the methodology, detailing the research design, algorithms used, and experimental setup. Chapter 5 covers project management aspects, while Chapter 6 evaluates the performance of the algorithms, while focusing on metrics such as planning time and success rate. Chapter 7 outlines the business strategy and will provide a commercialization plan for the integrated grasp and motion planning system. In this section we discuss target industries, potential business models, and industry partnerships. Finally, Chapter 8 summarizes the key findings, discusses the implications, and offers recommendations for future research.

# Chapter 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.

A close-up of a grid

Description automatically generated

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 single tree rooted at the start position by randomly sampling points in the configuration space and connecting them to the nearest node in the existing tree. It explores the space by extending the tree towards the goal, biased by the random samples. This makes RRT ideal for dynamic tasks that require real-time path planning (LaValle, 1998). Its adaptability, in contrast to PRM, made it a strong candidate for integrated grasp and motion planning.

Table 1 RRT and PRM key differences

|  |  |  |
| --- | --- | --- |
| Feature | RRT | PRM |
| Structure | Tree (single tree) | Graph (roadmap) |
| Use Case | Single-query | Multi-query |
| Exploration | Incremental exploration from start | Roadmap building, then querying |
| Reusability | Not reusable | Roadmap is reusable |
| Optimality | Not optimal (RRT\* is optimal) | Not optimal (PRM\* is optimal) |
| Strengths | Fast exploration, dynamic environments | Efficient in static, complex environments |
| Narrow Passages | Struggles in narrow spaces | Can handle narrow spaces with enough samples |
| Environment Focus | Dynamic, large environments | Static, reusable environments |

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.

## 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 Operations 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.

### 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 Operations

Before discussing the algorithms themselves, it is useful to define several primitive procedures 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 . If this new connection is valid (i.e., it avoids obstacles), the new point is added to the tree as a vertex, and the edge (v, ) 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 ), 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) to improve the algorithm's ability to find not just any valid path, but the most efficient one in terms of distance or cost. While RRT builds a tree by expanding the nearest node to a randomly sampled point in the configuration space, RRT\* introduces a critical optimization mechanism by rewiring the tree during expansion.

In RRT\*, when a new node is added to the tree, the algorithm does not just connect it to the nearest node x\_{nearest}. Instead, RRT\* evaluates all nearby nodes within a radius and rewires them if a lower-cost path can be found. This rewiring ensures that the tree grows not only to cover the space but also to optimize the path cost, typically in terms of distance.

This modification introduces two key changes:

1. **Radius-based Neighbor Search**: RRT\* expands by searching within a radius around each new node, rather than just connecting to the nearest one. This radius decreases as the tree grows, balancing exploration and optimization. The parameter controls the step size, limiting how far new nodes can extend, ensuring smoother connections and more efficient paths.
2. **Rewiring for Optimality**: After adding , RRT\* evaluates whether connecting to any nearby nodes can improve the total cost from the root to each node. If a better path is found, the parent-child relationships in the tree are updated (rewired) accordingly.

The cost function for RRT\* is generally the Euclidean distance between nodes, but it can be adjusted to reflect other task-specific objectives such as time or energy consumption. The algorithm is guaranteed to converge to an optimal solution as the number of iterations increases, though this comes at the cost of increased computational time compared to the original RRT.

Mathematical Formulation

1. **Connection Radius**: The radius for connecting to nearby nodes is defined as:  
   where is the number of nodes and is the dimensionality of the space. This radius decreases with , ensuring that connections are more localized as the tree grows.
2. **Cost Function**: For each node , the cost to reach that node is:  
     
      
     
   The parent of each node is rewired if a lower-cost path is found through another nearby node.
3. **Rewiring Condition**: After adding a new node , RRT\* rewires any nearby node if the path through reduces the overall cost:  
     
      
     
    If this condition holds, the edge from to is removed, and a new edge from to is added.

The pseudocode for the RRT\* algorithm (Karaman & Frazzoli, 2011) is as follows:

|  |
| --- |
| Algorithm 3: RRT\* |
| 1  2  3  4  5  6  7  8  9  10 // Rewiring  11  12  13  14 // Rewire the tree  15  16  17 |

### 2.2.6 JPplusRRT: Enhanced Speed and Adaptability

JPplusRRT, introduced by Vahrenkamp et al. (2009), attempts to balance exploration efficiency with faster decision-making in dynamic environments. Unlike RRT\*, which optimizes for path quality, JPplusRRT focuses on quickly finding feasible paths by integrating grasp planning directly into the RRT framework. By using the Jacobian pseudoinverse, as outlined in their paper, it provides more direct control over the movement of the robot's end-effector in task space, ensuring rapid adaptability to changes in the environment while maintaining task-specific goals like grasping.

The core of JPplusRRT involves steering the robot's configuration towards the desired grasping pose while continuously adjusting the robot’s joint configuration through Jacobian-based feedback. The pseudoinverse of the Jacobian matrix, , allows the algorithm to efficiently calculate how the robot should move in its configuration space to reach the desired pose in task space.

#### **Key Features of JPplusRRT:**

1. **Task-Space Guided Exploration:** Instead of only focusing on the configuration space (), JPplusRRT uses a task-space goal direction to steer the search.
2. **Jacobian Pseudoinverse:** The algorithm utilizes the pseudoinverse of the Jacobian matrix to compute joint-space movements that minimize the error between the current end-effector position and the goal.
3. **Multiple Goal Targets:** JPplusRRT incorporates a set of feasible grasps and iteratively selects a grasp as the target for the robot’s end-effector.
4. **Simplified Approach:** JPplusRRT avoids complex inverse kinematics (IK) calculations, making it faster and better suited for real-time scenarios, albeit with potentially suboptimal paths.

#### **Mathematical Formulation:**

* **Jacobian Pseudoinverse:** The pseudoinverse of the Jacobian matrix, , is used to calculate changes in joint angles based on the difference between the desired task-space position and the current position:

Where is the pseudoinverse of the Jacobian matrix, and is the difference between the current and target positions in task space.

* **Steering Function:** To move towards the goal, JPplusRRT calculates small steps in configuration space using the pseudoinverse of the Jacobian:

This approach ensures that the robot’s movement in C-space is guided by its task-space objectives (e.g., reaching a desired grasp pose).

* **Goal Bias:** The algorithm assigns a probability to bias the expansion towards the goal, ensuring that the search prioritizes reaching valid grasping poses.

#### **Pseudocode for JPplusRRT:**

|  |
| --- |
|  |
| 2.  3.  4.  5.  6.  7.  8.  9.  10.  Function:  1.  2.  3.  4.  5.  6.  7.  8.  9.  10.  11.  12.  13. |

### 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

Motion planning algorithms play a critical role in enabling robotic systems to navigate complex environments, solve high-dimensional planning problems, and execute tasks autonomously. As these algorithms have evolved, comparing their performance across multiple metrics has become essential to identifying which algorithms are best suited for specific tasks.

### 2.3.1 Criteria for Comparison: Planning Time, Success Rate, and Exploration

Several metrics are commonly used to evaluate the effectiveness of motion planning algorithms: **planning time**, **success rate**, and **exploration efficiency**.

* **Planning Time**: The time taken by an algorithm to generate a valid path from the start to the goal configuration is critical in real-time applications such as autonomous driving and robotics in dynamic environments. Rapidly Exploring Random Trees (RRT) and its variants, including RRT\* and P-RRT\*, are known for their rapid planning times, particularly in high-dimensional spaces (Qureshi et al., 2019). Algorithms like RRT are designed to quickly explore large spaces, providing a feasible solution in a relatively short time frame. However, speed often comes at the cost of path quality, as RRT does not prioritize finding optimal paths.
* **Success Rate**: This metric assesses the likelihood that an algorithm will successfully find a valid path within a given set of conditions. For example, **Probabilistic Roadmaps (PRM)** are known for their high success rates in static, multi-query scenarios, particularly in environments that allow for precomputed roadmaps to be reused (Kavraki et al., 1996). However, PRM struggles in dynamic environments where obstacles or goals might change during execution, leading to lower success rates. RRT\*, by contrast, improves upon RRT by refining paths as the tree expands, which generally increases success rates in more complex environments (Karaman & Frazzoli, 2011).
* **Exploration Efficiency**: This criterion refers to how well an algorithm explores the configuration space, especially in high-dimensional or cluttered environments. RRT is well-suited for environments requiring extensive exploration due to its incremental nature of sampling random points in the free configuration space (​) and attempting to connect them to the nearest existing node (LaValle, 2006). However, while RRT explores quickly, it often produces non-optimal paths. Variants like RRT\* enhance exploration by not only searching for feasible paths but also optimizing the tree to achieve more efficient routes (Karaman & Frazzoli, 2011). Exploration efficiency becomes particularly important when robots must navigate dynamic environments or handle complex objects in uncertain conditions, as highlighted in the work of Manzinger et al. (2021).

### 2.3.2 Testing in Shared Environments: Consistency and Parameters

To ensure a fair comparison of different motion planning algorithms, it is essential to test them in shared environments with consistent parameters. This approach enables researchers to assess how each algorithm performs under identical conditions, ensuring that any differences in performance are due to the algorithm itself and not external variables.

Standardized frameworks such as **Jogramop** (Rudorfer et al., 2024) provide pre-defined benchmark scenarios that mimic real-world challenges, such as navigating cluttered spaces or avoiding obstacles in environments like those seen in autonomous driving or robotics manufacturing. These benchmarks allow researchers to test algorithms like RRT, PRM, and Trajectory Planning methods (Choset et al., 2005) in environments with consistent obstacle density, path complexity, and dynamic elements. Additionally, frameworks such as **PyBullet** are often employed to simulate these environments and ensure that real-world parameters, like sensor inaccuracies and mechanical limitations, are accounted for during testing (Fan, 2023).

Moreover, parameter sensitivity is a critical aspect of testing. Algorithms may be highly sensitive to specific parameters such as step size, sampling rate, or search radius, all of which can significantly impact their performance. For example, RRT\* benefits from smaller step sizes, which allow the algorithm to explore paths more carefully and refine them toward optimality. On the other hand, PRM requires a sufficiently high sampling density to ensure that the roadmap connects the environment's critical regions effectively (Kavraki et al., 1996). Consistent testing environments provide valuable insights into how each algorithm performs under varied parameter settings.

### 2.3.3 Performance Insights

Studies have shown that RRT is highly effective in quickly generating feasible paths in high-dimensional configuration spaces, making it a suitable choice for dynamic or time-sensitive applications like robotic surgery and autonomous vehicles (LaValle, 2006; Fan, 2023). However, RRT’s lack of optimality is a significant drawback. As a response, RRT\* was developed to balance exploration and optimal path finding, significantly improving the quality of the paths at the cost of increased computational time (Karaman & Frazzoli, 2011).

PRM, on the other hand, excels in static, multi-query scenarios. Once the roadmap is built, it can be used repeatedly for different queries, offering high computational efficiency for applications where repeated planning is necessary, such as in automated warehouses or free-floating space robots (Liniger & Gool, 2020; Zhang & Zhu, 2020). However, PRM's performance declines in environments that are constantly changing, such as those with dynamic obstacles or moving goals (Kavraki et al., 1996).

Recent advancements have integrated machine learning techniques with traditional algorithms to enhance performance in more dynamic settings. For example, reinforcement learning has been employed to improve pathfinding strategies by enabling robots to learn from previous tasks and adapt to new environments (Tai et al., 2017; Kahn et al., 2018). Machine learning models trained on large datasets have significantly reduced planning times by predicting feasible grasps or paths based on past experiences (Mahler et al., 2017; Calandra et al., 2018). These hybrid approaches show promise in overcoming the limitations of traditional sampling-based methods, especially in environments where real-time adaptability is crucial.

RRT and PRM are effective in scenarios with few dynamic elements, while RRT\* and its derivatives perform better in tasks requiring a balance between exploration and path quality. The integration of learning-based methods provides new avenues for further enhancing motion planning algorithms, making them more adaptive and capable of handling real-time challenges (Ali & Lee, 2020; Kopicki et al., 2016).

## 2.4 Grasp Selection in Motion Planning

In the domain of robotics, selecting the appropriate grasp is an essential component of successful manipulation tasks. The ability to identify stable and task-specific grasps allows robots to perform complex manipulation in dynamic environments. This section provides an overview of grasp selection, highlighting its importance in integrated grasp and motion planning, and describes the key technical steps involved in the grasp planning process.

### 2.4.1 What is a Grasp? Definition and Importance

A grasp refers to the method by which a robot secures an object, ensuring stability and control over the object during manipulation tasks. The grasp is influenced by several factors, including the shape and size of the object, the robot’s end-effector, and the requirements of the specific task. The importance of selecting an appropriate grasp cannot be overstated, as it directly affects the robot's ability to manipulate objects successfully and safely in both controlled and dynamic environments.

The Force Closure and Form Closure theories are fundamental concepts that guide the definition of a good grasp. Force Closure ensures that the robot can resist external forces from any direction, achieving a stable grip through the application of balanced forces (Bicchi, 1995). This is critical in dynamic tasks where external disturbances can destabilize the object. On the other hand, Form Closure secures the object by positioning the robot's contact points to prevent any motion of the object, relying on geometric stability rather than applied forces (Mishra et al., 1987). These foundational theories have laid the groundwork for many grasp planning algorithms, influencing how robots determine where and how to grasp an object.

In practical applications, Task-Oriented Grasping shifts the focus from pure stability toward optimizing the grasp for the task at hand (Ciocarlie et al., 2009). For instance, when a robot needs to perform multiple actions with an object, such as picking it up, moving it, and placing it, the grasp must not only be stable but also facilitate these subsequent actions. Research by Prats et al. (2007) emphasizes this approach, using simplified geometric and structural descriptions of objects to select grasps that are tailored to the task requirements. This concept of task-specific grasping is especially important in fields like industrial automation and healthcare, where robots are required to perform precise and repetitive actions.

Moreover, advancements in grasp planning algorithms have expanded the scope of what robots can manipulate. Yan et al. (2019) describe approaches that address grasp stability for multi-fingered hands, enabling robots to perform more complex manipulation tasks. These systems analyse the geometry of objects and the configuration of robotic fingers to ensure a stable grasp, even for irregularly shaped items.

### 2.4.2 Grasp Planner: Selection Process and Technical Steps

A grasp planner is the mechanism by which a robot selects the most suitable grasp for a given object and task. The process of grasp selection involves multiple steps, from analyzing the object to computing a collision-free grasping pose. This process is critical for achieving seamless integration between grasp planning and motion planning.

The first step in the selection process is object analysis, which typically involves using sensors to capture the object's geometry, size, and surface properties. Techniques such as over-segmented meshes and the use of relational databases for object representation have been employed to improve grasp planning, particularly in complex regrasping tasks (Wan & Harada, 2017). These methods provide a detailed understanding of the object’s shape, allowing the grasp planner to identify potential grasp points that maximize stability and effectiveness.

Next, the planner must determine the best possible grasp configuration based on task requirements. Analytical approaches, such as those described by Ferrari & Canny (1992), use mathematical models to evaluate potential grasps by calculating the forces and torques that will be applied. This enables the planner to select a grasp that balances stability and task efficiency. Sampling-based methods further extend this by exploring different grasp configurations and selecting the one that optimizes specific criteria, such as stability or the ability to perform multiple tasks (Bohg et al., 2014).

Once potential grasps are identified, the planner moves to the collision checking and inverse kinematics (IK) stage. At this point, the robot evaluates whether the grasp configuration is feasible, ensuring that the robot’s arm can physically reach the object without colliding with obstacles. The IK-RRT algorithm, for example, integrates inverse kinematics into the motion planning process, solving both motion and grasping challenges simultaneously (Vahrenkamp et al., 2010). In this approach, the grasp planner ensures that not only is the grasp valid, but that the robot’s arm can achieve the required pose without violating environmental constraints.

Finally, the selected grasp is validated through simulation or real-world execution, where additional adjustments may be necessary to account for dynamic environmental factors or inaccuracies in sensory data (Calandra et al., 2018). The integration of multi-modal sensors, such as tactile and visual sensors, has further enhanced this stage by providing more accurate data on the object and surrounding environment (Kopicki et al., 2016). These advancements help robots dynamically adjust their grasps in response to changes, ensuring robustness in execution.

In summary, the grasp selection process involves detailed object analysis, evaluation of potential grasps based on task requirements, collision checking, and execution validation. As robots become more autonomous, grasp planners will continue to evolve, integrating more sophisticated machine learning models and real-time sensing to improve grasp performance in complex, real-world environments.

## 2.5 Other Algorithms in Grasp and Motion Planning

In addition to well-established algorithms like RRT and PRM, several other algorithms have emerged to address the challenges of integrated grasp and motion planning. These algorithms are designed to improve the efficiency, adaptability, and optimality of robotic manipulation tasks, particularly in environments that are complex or dynamic.

One of the key advancements in this area is the development of **Task-Oriented Grasping** algorithms. These algorithms, unlike traditional methods that focus solely on grasp stability, aim to optimize the grasp based on the specific task the robot needs to perform. By incorporating factors such as object manipulation and positioning into the planning process, task-oriented grasping algorithms provide more versatile and efficient solutions for industrial applications and service robots (Ciocarlie et al., 2009). Additionally, algorithms like **Grasp-RRT** have combined grasp planning with motion planning to generate collision-free trajectories toward optimal grasps, effectively streamlining the planning process (Vahrenkamp et al., 2010).

**Hierarchical Task Networks (HTN)** represent another significant contribution to grasp and motion planning. HTNs decompose complex tasks into smaller, more manageable sub-tasks, enabling robots to plan their actions hierarchically. This method has been particularly effective in multi-robot systems and environments where robots must coordinate with each other while avoiding collisions and optimizing their motion paths (Leu et al., 2022). Furthermore, **reinforcement learning,** and other machine learning approaches are increasingly integrated into grasp and motion planning. These methods leverage large datasets to enable robots to learn from experience and improve their performance over time (Tai et al., 2017). By learning from previous tasks, robots can generalize their grasp and motion strategies across different objects and environments, enhancing their adaptability.

**Convex optimization** and **nonlinear programming** have also gained prominence in grasp and motion planning algorithms. These methods allow for precise control over both the grasping and motion components, enabling robots to generate optimal paths and grasps in real time (Garrett et al., 2021). These optimization-based approaches are particularly useful in scenarios where multiple criteria, such as stability, speed, and energy consumption, must be balanced.

In summary, these advanced algorithms have expanded the capabilities of robotic systems by addressing the limitations of earlier methods. They enhance the robot’s ability to plan both grasping and motion simultaneously, ensuring more efficient and reliable performance in diverse tasks and environments.

## 2.6 Gaps in literature

Despite significant advancements in integrated grasp and motion planning, several gaps remain in the existing literature that must be addressed to achieve greater progress in the field.

One of the key gaps lies in **scalability**. Many algorithms currently in use struggle to scale efficiently to high-dimensional tasks, particularly in environments filled with dynamic obstacles. For instance, while sampling-based methods such as RRT and PRM are effective in lower-dimensional spaces, they tend to become computationally expensive and inefficient in highly complex environments where many degrees of freedom must be considered (Kavraki et al., 1996). This issue is especially prevalent in multi-robot systems, where coordination and collision avoidance increase the computational load exponentially (Yu & LaValle, 2016).

Another significant gap is in **real-time adaptability**. While some algorithms, such as RRT, can quickly find a feasible path, their adaptability to dynamic environments is often limited. As environments change—whether due to moving obstacles, shifting goals, or external disturbances—many algorithms lack the capacity to recompute paths efficiently enough to ensure smooth operation (Kopicki et al., 2016). Real-time motion planning algorithms, such as Model Predictive Control (MPC), offer some solutions, but even these approaches struggle with scalability and high-dimensional spaces (Falcone et al., 2007).

**Generalization to novel tasks** is another persistent challenge. Many current methods rely on precomputed data or fixed assumptions about the environment, making them less effective in handling unfamiliar or dynamically changing situations (Mahler et al., 2017). This limitation is particularly problematic for robots operating in unstructured or semi-structured environments, such as homes or hospitals, where the diversity of objects and scenarios cannot be anticipated in advance. Although machine learning approaches such as deep learning and reinforcement learning have made strides in enabling robots to learn from past experiences and improve their adaptability, much work remains to be done to ensure that robots can handle truly novel tasks with minimal human intervention (Calandra et al., 2018).

Finally, **computational demands** pose a significant hurdle to the wider adoption of advanced motion planning algorithms. The need for real-time performance in applications such as autonomous driving, surgical robotics, or industrial automation places immense pressure on algorithms to be both fast and accurate. Balancing these two requirements often leads to trade-offs that limit the applicability of these algorithms in real-world scenarios (Ziegler et al., 2014).

Addressing these gaps will require continued research into developing more scalable, adaptable, and generalizable planning algorithms. Future efforts should focus on integrating machine learning with traditional planning methods, improving computational efficiency, and expanding the ability of robots to generalize to novel tasks and environments.

## 2.7 summary

This chapter reviewed the key concepts, algorithms, and challenges in integrated grasp and motion planning, a critical area of robotics that enables autonomous systems to perform complex manipulation tasks in dynamic and uncertain environments. We began by discussing the fundamental principles of grasp selection, including theories such as Force Closure and Task-Oriented Grasping, which guide how robots determine stable and efficient grasps for various tasks (Bicchi, 1995; Ciocarlie et al., 2009). Following this, we explored the technical steps involved in grasp planning, from object analysis to collision checking and execution validation, emphasizing the importance of integrating grasp and motion planning for optimal performance (Ferrari & Canny, 1992; Vahrenkamp et al., 2010).

The chapter also provided a comparative analysis of motion planning algorithms such as RRT, PRM, and their variants, evaluating their performance based on criteria like planning time, success rate, and exploration efficiency (Karaman & Frazzoli, 2011; LaValle, 2006). While these algorithms have enabled significant progress in autonomous robotic systems, key gaps remain in terms of scalability, real-time adaptability, generalization to novel tasks, and computational efficiency (Kavraki et al., 1996; Kopicki et al., 2016). Future research will need to focus on addressing these limitations to unlock the full potential of integrated grasp and motion planning in real-world applications.

The chapter concludes by identifying the ongoing challenges in this field and suggesting that future advancements will likely arise from integrating machine learning approaches with traditional planning methods to improve scalability, adaptability, and generalization capabilities. These advancements will be crucial for enabling robots to perform complex manipulation tasks autonomously and efficiently in a wide range of industries.

# Chapter 3: Problem Description

#### 3.1 Integrated Grasp and Motion Planning

Robotic systems face significant challenges when tasked with grasping and manipulating objects in dynamic environments. Traditionally, grasp planning and motion planning have been handled separately, which can lead to inefficiencies and suboptimal performance. In many scenarios, the interaction between the grasp and the robot’s motion path is critical for achieving smooth and efficient task execution. Integrated grasp and motion planning approaches seek to resolve this by simultaneously planning how the robot grasps an object and how it moves while avoiding obstacles and maintaining task feasibility.

This research addresses a specific challenge in the domain of integrated grasp and motion planning. Specifically, it focuses on evaluating different path planning algorithms to determine which methods can best optimize both the robot’s grasp and motion, particularly in constrained or complex environments.

#### 3.2 Problem Statement

The core problem this research addresses is the need for efficient and reliable integrated grasp and motion planning algorithms for robotic manipulators. This is especially challenging when operating in environments with varying object configurations and dynamic constraints. The primary question driving this research is how different algorithms, particularly RRT\*, perform in solving these planning tasks, compared to other established algorithms like J+RRT and IKRRT.

In this context, the problem can be broken down into the following components:

1. **Grasp Selection**: The robot must determine a stable and feasible grasp configuration for each object in the workspace. This involves analyzing possible grasp poses and selecting one that maximizes stability while being feasible in terms of the robot's kinematics.
2. **Motion Planning**: Once a grasp is selected, the robot must plan a path from its initial configuration to the object, ensuring that it avoids obstacles and operates within its physical limitations.
3. **Dynamic Environments**: In real-world applications, the environment may change dynamically, making it necessary for the algorithms to adapt to these changes while maintaining efficiency and success in task completion.

The objective of this research is to evaluate how well the RRT\*, JPlusRRT, IKRRT, and BIKRRT algorithms handle integrated grasp and motion planning, particularly in complex environments with diverse object placements and obstacles.

#### 3.3 Specific Challenges

The specific challenges addressed by this dissertation are as follows:

1. **High-dimensional Search Space**: The configuration space of robotic manipulators, especially those with six or more degrees of freedom, is vast and difficult to search efficiently. The challenge is to evaluate how RRT\* and other algorithms manage this space while considering both grasping and motion.
2. **Cluttered and Complex Environments**: Real-world environments are often cluttered and dynamic, which makes the planning process even more difficult. This research tests the robustness of the algorithms in these environments, particularly those with confined spaces and irregular object arrangements.
3. **Algorithm Performance**: The goal is to evaluate the trade-offs between computational efficiency (planning time), success rates (finding a feasible path), and the quality of the paths generated by each algorithm.
4. **Practical Implementation**: While many algorithms have theoretical benefits, practical implementation often reveals constraints or bottlenecks that were not anticipated. By implementing these algorithms in the PyBullet environment and the Jogramop framework, this research aims to identify these practical challenges.

#### 3.4 Importance of the Problem

The success of integrated grasp and motion planning is crucial in advancing the capabilities of robotic systems. Robots are increasingly being deployed in industries such as manufacturing, healthcare, and service sectors, where they need to handle complex tasks with high reliability. The findings of this research will contribute to improving the performance of robotic systems, making them more adaptable and efficient in real-world applications.

#### 3.5 Summary

This chapter outlines the core problem that this dissertation addresses: the challenge of efficient and robust integrated grasp and motion planning in dynamic environments. The goal is to evaluate several algorithms, with a particular focus on RRT\*, to determine their strengths and limitations in handling complex robotic manipulation tasks. The following chapters will detail the methodology and experimental setup used to tackle this problem and present the findings from the evaluations of the different algorithms.

# Chapter 4: Methodology

## 4.1 Introduction

The primary objective of this chapter is to detail the implementation of several motion planning algorithms—RRT, RRT\*, JPlusRRT, IK-RRT, and BIK-RRT—using Python, and to evaluate their performance in both PyBullet and the Jogramop framework. These algorithms were chosen to explore their applicability to integrated grasp and motion planning tasks, particularly in robotic environments with dynamic constraints and complex object configurations.

The experimental process is divided into two phases. The first phase focuses on implementing and testing the algorithms in the PyBullet environment, where their core performance metrics, such as planning time and success rate, were analyzed in simpler scenarios. In the second phase, the RRT\* algorithm was tested within the Jogramop framework, which offers standardized benchmark scenarios designed to challenge the algorithms with more complex and confined environments.

This chapter will cover the research design, the rationale for the algorithms and frameworks used, the experimental setup, the evaluation metrics, and the presentation of the results. By conducting these experiments, we aim to demonstrate how the algorithms perform in practical robotic applications and compare their effectiveness in solving integrated grasp and motion planning problems.

## 4.2 Problem Formulation

The problem addressed in this research involves the integrated grasp and motion planning for a robotic manipulator, with a focus on comparing algorithmic performance in benchmark scenarios.

In this context, we consider a set of potential grasps, where each grasp consists of a grasp pose representing the 6-DoF position and orientation of the gripper relative to the object.

The motion planning component involves searching the robot's configuration space , where each configuration represents a particular set of joint angles for the manipulator. The collision-free subset of the configuration space is denoted as . The objective for the motion planner is to find a path from an initial configuration to a goal configuration , ensuring that the chosen grasp ​ is reached while minimizing path length, avoiding obstacles, and adhering to the robot's kinematic constraints (Rudorfer et al., 2024).

Figure 1 Various scenarios from the Jogramop framework showcasing a robot hand attempting to grasp different objects.

To compare algorithms, we set up benchmark scenarios involving a variety of objects, grasps, and environmental configurations. The goal is to determine how well each algorithm performs in terms of path planning efficiency, and adaptability to dynamic changes in the environment. Performance metrics such as planning time, and success rate will be used to evaluate the effectiveness of each algorithm in real-time simulations.

## 4.3 Research Design

**Research Methodology**

The research follows a **quantitative experimental methodology** aimed to implement multiple motion-planning algorithms in a controlled simulation environment. The experiments were conducted to compare the algorithms based on key metrics such as planning time, success rate, and the exploration of the configuration space. The study is divided into two distinct phases, each designed to test the algorithms under different environmental conditions and complexity levels.

**Phases of Research**

1. **Phase 1**:  
   The first phase involved the implementation and testing of **RRT, RRT\***, **JPlusRRT, IK-RRT, and BIK-RRT** in the **PyBullet environment**. This phase was designed to assess the core performance of each algorithm in relatively simpler scenarios, allowing for a direct comparison of their effectiveness in solving motion planning problems. The focus was on how each algorithm explores the configuration space, generates feasible paths, and handles dynamic constraints.
2. **Phase 2**:  
   In the second phase, the **RRT\*** algorithm was implemented in the **Jogramop framework**, which features 20 standardized benchmark scenarios. These scenarios are more complex and confined than those in Phase 1, providing a more structured environment to evaluate RRT\*'s performance under more challenging conditions. The focus in this phase was on testing the algorithm's robustness and optimality in navigating obstacle-rich environments.

**Environment Setup**

* **PyBullet Environment**:  
  The PyBullet environment provided a simpler testing ground for initial comparisons between the algorithms. A 6-DOF Franka Panda robotic arm was used, interacting with basic objects and obstacle configurations. This environment allowed for testing in controlled, repeatable conditions, focusing on key performance metrics such as path efficiency and collision avoidance. The flexibility of PyBullet facilitated the testing of various object sizes and obstacle densities, ensuring comprehensive performance evaluation in the first phase.
* **Jogramop Framework**:  
  The Jogramop framework (Rudorfer et al., 2024) was utilized in the second phase to provide a more structured and standardized set of benchmark scenarios. These 20 predefined scenarios simulate real-world challenges, including confined spaces and complex obstacle distributions, making it an ideal environment for testing the more advanced features of RRT\*, such as path optimality and rewiring. The Jogramop framework allows for repeatable testing under uniform conditions, enabling more precise performance comparisons.

**Control Measures**

To ensure a fair comparison between algorithms, several control measures were implemented:

* **Consistent Parameters**: Key parameters such as step size, goal bias, and the number of iterations were kept consistent across all algorithms to eliminate bias in the comparison process. This ensured that any performance differences were due to the inherent capabilities of the algorithms rather than external factors.
* **Repetition of Trials**: Each experiment was repeated 100 times for each scenario, allowing for statistically significant results. Repeated trials helped minimize the effects of outliers and variability in performance.
* **Controlled Environment**: Both the PyBullet and Jogramop environments were designed to minimize external variability. Environmental factors such as object placement, obstacle density, and configuration space boundaries were kept constant throughout the experiments to ensure that all algorithms were tested under identical conditions.

## 4.4 Algorithms and Frameworks

The selection of algorithms and simulation frameworks in this research was guided by the need to explore various aspects of integrated grasp and motion planning, from basic pathfinding to handling complex constraints. Each algorithm and framework was chosen to fulfill specific roles in the overall implementation and testing process.

* **RRT (Rapidly Exploring Random Tree)**:  
  RRT was selected as a **baseline algorithm** for motion planning. As a widely used sampling-based method, it incrementally builds a tree that explores the configuration space quickly. RRT is especially useful for understanding how basic motion planning operates in high-dimensional spaces, making it an ideal starting point for implementing more advanced algorithms.
* **RRT\* (RRT-Star)**:  
  RRT\* was chosen due to its **path optimization capabilities**. Unlike RRT, RRT\* guarantees asymptotic optimality by refining paths as it grows the tree. Its ability to minimize path cost and produce more efficient trajectories makes it essential for tasks where grasping and motion planning must be optimized simultaneously. This algorithm is particularly well-suited for scenarios involving confined spaces or complex object configurations.
* **JPlusRRT, IK-RRT, and BIK-RRT**:  
  These algorithms were selected to address the **more complex aspects of grasp and motion planning**.
  + **JPlusRRT** integrates the Jacobian pseudo-inverse, enabling the robot to handle kinematic constraints more efficiently, which is critical in grasp planning.
  + **IK-RRT** incorporates **inverse kinematics** to ensure that the robot can reach desired grasp configurations while adhering to the robot's joint limits. This makes it suitable for motion planning tasks where the robot’s joint configurations play a key role.
  + **BIK-RRT** further extends the capabilities of IK-RRT by introducing **bidirectional search**, allowing the algorithm to explore the configuration space from both the start and goal configurations. This method enhances the algorithm’s ability to find feasible paths in complex environments, particularly where direct paths are difficult to find.

**Frameworks**

The choice of simulation frameworks—**PyBullet** and **Jogramop**—was motivated by their ability to support the experimental objectives at different stages of the project.

* **PyBullet**:  
  PyBullet was chosen for its **flexibility and ease of use** in implementing and testing motion planning algorithms. It supports real-time physics simulations and provides a wide range of tools for testing robotic systems. For this research, PyBullet was used in the initial phase of testing to implement and verify the basic functionality of the algorithms in simpler scenarios. It allowed for rapid prototyping, debugging, and visualization of the algorithms' behavior in a controlled environment.
* **Jogramop Framework**:  
  The **Jogramop framework** was selected for the second phase of testing due to its **structured benchmark scenarios**. Jogramop offers 20 predefined scenarios, designed specifically to challenge motion-planning algorithms in environments with more complex obstacles and confined spaces. This framework allowed for a more rigorous exploration of the RRT\* algorithm in standardized conditions, making it ideal for testing more advanced capabilities like path optimization and obstacle avoidance. The uniformity of these scenarios ensured consistent testing and helped in comparing the algorithm’s performance across different conditions.

**Algorithm-Specific Justifications**

* **RRT** was selected for its ability to quickly explore large configuration spaces, providing a good baseline for understanding basic motion planning behavior. Its simplicity makes it a valuable starting point for testing and implementing more complex algorithms.
* **RRT\*** was chosen to extend the exploration process by incorporating path optimization. Its ability to find optimal paths, particularly in cluttered environments, makes it an essential component for tasks where minimizing the trajectory’s cost is important.
* **JPlusRRT, IK-RRT, and BIK-RRT** were selected to investigate how grasp and motion planning can be effectively combined. These algorithms address specific challenges in robotic manipulation, such as managing kinematic constraints and finding feasible grasps. Their inclusion in the research allows for a more detailed analysis of how motion planning algorithms can handle real-world complexities in grasp planning.

In summary, the combination of these algorithms and frameworks allowed for a comprehensive exploration of motion planning in both simple and complex environments. PyBullet provided a flexible environment for initial implementation, while the Jogramop framework ensured that the algorithms could be rigorously tested in standardized scenarios.

## 4.5 Experimental Setup

#### 4.5.1 Simulation Environments

The experiments were conducted in two distinct simulation environments to implement and test the algorithms:

* **PyBullet**: In the first phase, the algorithms—RRT, RRT\*, JPlusRRT, IK-RRT, and BIK-RRT—were implemented in PyBullet, a real-time physics simulation engine. A 6-DOF Franka Panda robotic arm was used for motion and grasp planning tasks. The scenarios tested in this environment were relatively simple, focusing on basic object manipulation, obstacle avoidance, and path planning tasks. This setup allowed for flexibility and quick iterations, making it ideal for initial algorithm testing and development.
* **Jogramop Framework**: In the second phase, the RRT\* algorithm was tested using the Jogramop framework, which includes 20 standardized benchmark scenarios. These scenarios are designed to introduce more complex and confined environments, including narrow passages and high obstacle densities. This provided a more structured testing environment where the algorithm’s ability to handle real-world motion-planning challenges could be explored.

#### 4.5.2 Testing Parameters

Several key parameters were adjusted during the experiments to suit the nature of the algorithms and scenarios:

* **Step Size (η)**: The step size, which controls how far the robot moves at each iteration, was adjusted based on the complexity of the scenario. Larger step sizes were used in simpler environments to allow faster path exploration, while smaller step sizes were applied in confined spaces to ensure smooth and precise paths.
* **Goal Bias**: This parameter increases the likelihood of sampling near the goal, improving convergence. In complex, confined spaces, a higher goal bias was set to help the algorithms quickly reach the goal.
* **Rewiring Radius**: For RRT\*, the rewiring radius was adjusted to optimize path cost. In environments with high obstacle density, a larger radius allowed the algorithm to rewire paths for greater optimization.

These parameters were fine-tuned based on the environment's complexity to ensure optimal algorithm performance while maintaining consistency across experiments.

#### 4.5.3 Testing Scenarios

The algorithms were tested in a variety of scenarios, ranging from simple environments to more complex obstacle-rich setups:

* **PyBullet Scenarios**: These scenarios involved grasping single or multiple objects, navigating around basic obstacles like walls or blocks, and planning collision-free paths. These scenarios were designed to test the foundational behavior of each algorithm.
* **Jogramop Scenarios**: The Jogramop framework provided more challenging environments, including narrow corridors, high obstacle densities, and confined spaces. These scenarios simulated real-world constraints where precision and optimization in path planning were critical.

#### 4.5.4 Repetitions and Trials

To ensure the reliability of the results, each algorithm was tested across **100 trials per scenario**. This repetition helped minimize the impact of variability or outliers in the data. During each trial, the system logged key metrics such as planning time and success rate, and the results were averaged to provide a clear comparison of performance across different algorithms and environments.

## 4.6. Metrics for Evaluation

The performance of the algorithms was evaluated using two key metrics:

* **Path Length:** This measures the total distance travelled from the start to the goal. A shorter path length indicates more efficient movement and optimization of the trajectory, especially important in constrained environments.
* **Planning Time:** This reflects the algorithm's computational efficiency. A shorter planning time indicates faster computation of a collision-free path.
* **Success Rate:** This measures the reliability of the algorithm in successfully finding a path to the goal. A high success rate indicates robustness in handling dynamic and obstacle-rich environments.

These metrics were chosen because they directly reflect the real-world performance of grasp and motion planning algorithms. Planning time assesses the computational cost, while the success rate measures the practical applicability of the algorithm in robotic tasks.

## 4.7 Overview of Results

The experimental setup was designed to assess the performance of four algorithms—JPplusRRT, RRT, IK-RRT, and BIK-RRT—across different scenarios in the PyBullet environment, while RRT\* was exclusively tested in the Jogramop framework. In these trials, the robot’s task was to reach one of three goal positions: center, left, and right, placed either on top of the table or under the table. The evaluation focused on key metrics: path length, planning time, and success rate, providing insights into each algorithm's exploration efficiency, speed, and robustness in handling various complexities.

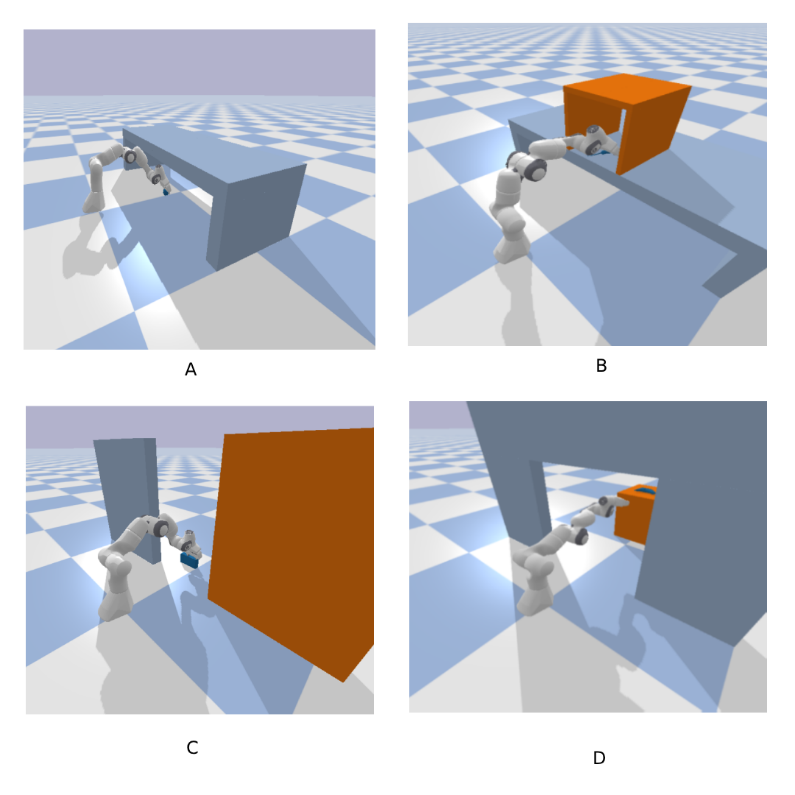
The experiments were divided into two primary environments:

**PyBullet environment:** Simpler scenarios were tested with three goal positions on top of or under a table.

A robot arm and a blue table

Description automatically generated with medium confidence  
**Jogramop framework:** More complex benchmark scenarios (20 in total) were progressively tested with RRT\*, where the complexity of the environment increased, especially with tighter obstacles and more constrained paths.

Table 1 Compare of average time after 100 trials of RRT Star algorithm in 20 different scenarios



### PyBullet Environment Results



**JPplusRRT:**

* This algorithm demonstrated the fastest planning times across almost all scenarios in the PyBullet environment, particularly excelling when the goals were placed on top of the table. Path lengths ranged between 0.3430 and 0.3487, indicating a fast exploration capability with reasonable paths.
* Success rates remained consistently high, at 95-97%, with only a slight dip in scenarios where goals were placed under the table. The more constrained environments slightly affected its speed, but the algorithm still maintained its robustness and adaptability to dynamic conditions.

**RRT\*:**

* As expected, RRT\* optimized for path quality, producing the shortest paths when navigating under the table. Path lengths ranged between 0.2580 and 0.4125, reflecting its focus on path optimization even in more constrained spaces.
* Planning times were higher, ranging between 0.0010 and 0.0062 seconds, particularly in more complex environments. The success rate dropped slightly, between 90-93%, in tighter spaces under the table, where the algorithm occasionally struggled to find a feasible path in the allowed iterations.

**IK-RRT:**

* IK-RRT demonstrated longer planning times, particularly in obstacle-rich environments under the table, with planning times ranging between 0.4427 and 0.4741 seconds. The added complexity of solving inverse kinematic constraints contributed to its slower performance.
* Despite this, IK-RRT maintained a high success rate of 98-99%, showing strong reliability in finding valid paths, albeit with longer path lengths compared to JPplusRRT and RRT\*.

**BIK-RRT:**

* BIK-RRT exhibited the longest paths and most inconsistent planning times, particularly in complex environments. In scenarios with the goal under the table, planning times fluctuated significantly, reaching up to 0.20 seconds in some cases.
* The success rates ranged from 91-94%, indicating variability in handling constrained environments. BIK-RRT's bidirectional search often led to over-exploration, resulting in longer paths and less consistent performance overall.

### Jogramop Framework Results: 20 Benchmark Scenarios

The Jogramop framework provided a more challenging environment, where RRT\* was tested across 20 benchmark scenarios. These scenarios gradually increased in complexity, with the addition of narrow passages, tight spaces, and complex obstacle configurations.

* Path Length: In the Jogramop environment, RRT\* excelled at generating short and optimized paths, particularly in moderately complex scenarios. Path lengths in this environment ranged between 0.2523 and 0.4125, reflecting the algorithm’s ability to consistently refine paths.
* Planning Time: As the complexity of the scenarios increased, particularly in environments with narrow passages or goals under the table-like structures, planning times increased slightly, with values ranging from 0.0010 to 0.0062 seconds. The focus on optimizing path quality led to longer computation times, especially in highly constrained environments.
* Success Rate: RRT\* maintained high success rates in simpler scenarios, but in the most challenging cases (e.g., tight corridors or narrow goals), the success rate dropped to 90-93%. This decline was primarily observed in scenarios where the algorithm struggled to find valid paths before reaching the maximum iteration count.

**Summary of Key Insights**

* JPplusRRT consistently demonstrated faster planning times, particularly in simpler environments, making it ideal for dynamic scenarios where quick adaptation is critical. However, its success rate dipped slightly in tighter spaces.
* RRT\*, while slower in terms of planning time, produced the shortest paths in both the PyBullet and Jogramop environments. The trade-off between path quality and computation time was evident, particularly in more complex environments.
* IK-RRT showed longer planning times but maintained the highest success rates across both environments, making it highly reliable for complex tasks that require precision in path planning.
* BIK-RRT was the most variable, with inconsistent performance in terms of both path length and planning time, especially in highly constrained environments. Its bidirectional search occasionally led to over-exploration, reducing its overall efficiency.

These results highlight the trade-offs between speed, path quality, and robustness across the different algorithms. JPplusRRT is best suited for rapid, dynamic environments, while RRT\* excels when path quality is the priority. IK-RRT offers reliability for more complex configurations, and BIK-RRT struggles with consistency but remains an option in bidirectional search contexts.

**4.7 Experimental Results**

* **Presentation of Results**:
  + Present the key results from both the PyBullet environment and Jogramop framework.
  + Use tables, figures, or graphs to illustrate differences in planning time, success rate, and exploration efficiency between the algorithms.
* **Result Comparison**: Compare the results of RRT, RRT\*, JPlusRRT, IK-RRT, and BIK-RRT in each scenario.
  + Highlight where each algorithm performed best or worst.
  + Discuss specific scenarios where certain algorithms outperformed others.

**4.8 Discussion of Findings**

* **Analysis of Results**:
  + Analyze the experimental results in detail.
  + Discuss how each algorithm's performance aligns with its theoretical strengths and weaknesses.
  + Highlight trends in planning time and success rates across different scenarios.
* **Algorithm Strengths and Weaknesses**:
  + For each algorithm, discuss where it excels and where it struggles (e.g., RRT\* performs well in complex environments but requires more computational resources).
* **Impact of Environment**: Discuss how the environment (e.g., PyBullet vs. Jogramop) impacted the results.
  + Was RRT\* more suitable for complex benchmark scenarios?
  + How did IK-RRT and BIK-RRT handle grasp constraints in the simpler environment?

## 4.9. Data Collection

Data was collected for each scenario by logging the planning time and success rate after every trial. The results were stored using automated logging scripts within the simulation framework, ensuring accurate tracking of each run. Each trial was repeated 100 times to minimize the effect of outliers, and any anomalous results were carefully reviewed and validated for consistency.

## 4.9. Challenges and Adaptations

During the research, several significant challenges arose, particularly in setting up the simulation environments and implementing the algorithms. Designing environments that closely mirrored real-world tasks was both time-consuming and technically demanding, as it required the careful tuning of parameters and extensive testing to ensure the accuracy and robustness of the simulations. Furthermore, the implementation of all the algorithms within the Jogramop framework turned out to be more complex than initially anticipated. JPlusRRT, in particular, posed difficulties due to its intricate planning and optimization requirements, demanding careful attention to both memory efficiency and computational complexity.

The original goal was to implement four distinct algorithms in the Jogramop framework. However, due to time constraints and the complexity of the task, only the RRT\* algorithm was fully implemented and tested. The adaptation of this plan allowed for a more focused comparison between RRT\*, JPlusRRT, and IK-RRT, the latter two of which had already been integrated into the framework. This shift in focus highlighted both the strengths and limitations of the RRT\* algorithm.

One of the most challenging aspects of the project was running RRT\* across 20 different scenarios, each tested 100 times. This involved substantial computational effort and highlighted the high computational cost of RRT\*, especially when scaling to more complex environments. The algorithm required careful configuration, tuning of parameters such as step size and goal direction probability, and constant monitoring to ensure the results were consistent. Balancing this level of computational demand with the practical limitations of hardware and time was a significant hurdle in the project, and it underscored the importance of computational efficiency in algorithm design and simulation testing.

# Chapter 5: Project Management

Effective project management was crucial to the successful implementation of my dissertation, which involved coding, testing, and comparing multiple path-planning algorithms in simulated environments. The project was managed by setting clear milestones, tracking progress, prioritizing tasks, and adapting to challenges that arose during the process.

A screenshot of a graph

Description automatically generated

Figure 3 Gantt Chart for Dissertation on Integrated Grasp and Motion Planning.

#### Timeline and Milestones

I began by defining specific milestones to guide the progress of the project. Initially, I implemented a simple RRT algorithm in a PyBullet simulation environment using a 6-DOF robot, the Franka Panda robot. As I gained more familiarity with the environment, I expanded to implementing advanced algorithms such as RRT\*, JPlusRRT, IKRRT, and BIKRRT. These milestones helped to keep the project structured, with a clear focus on building and testing the algorithms sequentially.

Later, I set up the Jogramop framework environment developed by Rudorfer et al., which included 20 benchmark scenarios. While the original goal was to implement multiple algorithms in this environment, due to time constraints, only RRT\* was fully implemented and compared with the benchmark results. Each phase of the project was organized and executed according to a timeline, and the overall progress was tracked to ensure timely completion of key tasks.

#### Task Management

The project was divided into three key areas: coding, testing, and evaluation of algorithms. The priority was to implement the RRT\* algorithm and its variations. Once the core algorithms were in place, I focused on testing their performance in different environments, specifically using the benchmark scenarios from the Jogramop framework.

Given the complexity of the task, coding and algorithm implementation took the majority of my time. Testing and evaluation followed naturally, with each iteration requiring detailed analysis and adjustments to ensure the algorithm's accuracy and performance. Task management followed an agile methodology, where iterative development cycles allowed for frequent testing and feedback.

#### Resource Management

I relied on PyBullet for simulations and used the Jogramop framework for comparing RRT\* with other algorithms in benchmark scenarios. GitHub was utilized for version control, while Trello was employed for project management and task tracking. Weekly meetings with my supervisor provided valuable guidance, helping me to resolve issues related to environment setup, algorithm implementation, and scenario testing.

The main limitation was the lack of comprehensive online resources for working with specific environments and algorithms, which added complexity to the project. Despite these challenges, having the jogramop framework, which provided ready-made environments, significantly improved the process, allowing me to focus on algorithm implementation and performance testing.

#### Risk Management

There were several risks and challenges during the project, particularly with algorithm performance and simulation failures. Setting up the simulation environment and ensuring compatibility with each algorithm required significant effort. Some algorithms did not perform as expected in certain environments, leading to several iterations of debugging and refinement.

To mitigate these risks, I reduced the number of algorithms tested in the new scenarios and focused on the performance of RRT\* in the jogramop framework. This allowed me to maintain the quality and depth of the analysis while still comparing the algorithm against benchmark scenarios. My supervisor’s guidance was instrumental in navigating these challenges.

#### Time Management

The most time-intensive phase of the project was coding and implementing the algorithms, which required careful attention to detail and frequent adjustments. I had initially planned to implement multiple algorithms in the new scenarios, but time constraints only allowed for the completion of RRT\*. As a result, I focused on ensuring that the RRT\* algorithm was thoroughly tested and compared to the benchmark scenarios in terms of runtime and success rate.

Despite these delays, the project was managed effectively, with time allocated to each phase based on its complexity and importance. Weekly progress tracking helped to keep the project on schedule, although some adjustments were made in the final weeks to accommodate additional testing.

#### Collaboration and Supervision

I maintained regular communication with my supervisor, Dr. Martin Rudorfer, meeting with him weekly or bi-weekly to discuss progress and receive feedback. His guidance was invaluable, particularly in helping me navigate challenges related to algorithm implementation and the setup of the simulation environment. Based on his input, I made several adjustments to the project plan, including the decision to focus on RRT\* in the benchmark scenarios due to time constraints.

The collaborative feedback loop allowed me to refine my approach continuously and make data-driven decisions throughout the project. My supervisor's support ensured that I stayed on track and met the key milestones necessary for the successful completion of the dissertation.

#### Challenges and Adaptations

One of the biggest challenges I encountered was setting up the simulation environments and frameworks required for algorithm testing. Implementing the RRT\* algorithm and others in these environments was not a straightforward task due to the need for custom configurations and adjustments. Additionally, comparing the results with the benchmarks required detailed attention to each scenario's setup.

To overcome these challenges, I focused on the jogramop framework, which provided pre-existing environments for testing, reducing the time spent on setup and allowing me to concentrate on algorithm performance. However, I had to adjust my original plan by limiting the number of algorithms tested in new scenarios, which allowed me to complete the project within the given timeframe.

In conclusion, effective project management played a crucial role in the successful execution of this dissertation. By setting clear milestones, managing tasks effectively, and mitigating risks, I was able to complete the implementation and testing of the RRT\* algorithm and compare its performance against established benchmarks. Despite some challenges, regular supervision and adaptive time management ensured the project's success.

# Chapter 6: Business Strategy and Market Integration

## Introduction to Business Strategy

In today’s rapidly evolving technological landscape, the commercialization of innovations requires a strategic approach that extends beyond product development. Business strategy plays a critical role in turning technological breakthroughs into marketable solutions by aligning technical capabilities with market needs, competitive positioning, and financial viability. For robotics and AI-driven solutions like integrated grasp and motion planning, a strong business strategy ensures that the product not only meets industry requirements but also differentiates itself from competitors. This involves identifying the right markets, establishing competitive advantages, forming industry partnerships, and developing a robust go-to-market plan. Without a clear strategy, even the most advanced technologies can struggle to gain traction, scale, and generate sustainable revenue. Therefore, the integration of business strategy is essential for maximizing the commercial potential of these innovations.

## Robotics Marketplace and Its Growth Potential

The robotics industry has experienced exponential growth over the past decade, driven by advancements in automation, AI, and machine learning. According to industry reports, the global robotics market is expected to reach over $75 billion by 2025, fuelled by increasing demand for industrial robots, autonomous systems, and AI-driven solutions in sectors such as manufacturing, healthcare, logistics, and consumer services. With industries seeking to optimize efficiency, reduce operational costs, and address labour shortages, the demand for sophisticated robotic systems capable of performing complex tasks, including integrated grasp and motion planning, continues to rise.

Key growth areas include collaborative robots (cobots) for flexible manufacturing processes, medical robots for precision surgeries, and autonomous mobile robots (AMRs) for logistics and warehousing. These trends underscore the immense potential for robotics innovations to capture a significant share of this expanding market, particularly as industries continue to automate complex and dynamic tasks.

## Market Landscape and Competitive Analysis

The current robotics market is characterized by a wide range of players offering various solutions for automation, robotics, and AI. Major industry players include companies like ABB, Fanuc, KUKA, and Boston Dynamics, each of which has established strong footholds in sectors such as industrial automation, logistics, and advanced robotics. Additionally, startups and research institutions are continually innovating with niche solutions that cater to specific challenges, such as grasp planning and motion optimization.

When it comes to integrated grasp and motion planning, some technologies focus solely on either grasp optimization or motion pathfinding, leaving a gap in fully integrated solutions that can handle both tasks in complex environments. Competitors may include companies developing autonomous robotic arms, automated material handling systems, and surgical robots, many of which use motion planning but lack advanced, real-time grasp integration.

**Potential Competitors:**

* **Robotiq:** Specializes in robotic grippers and collaborative robots but focuses on grasping without motion planning integration.
* **Universal Robots:** Offers cobots with advanced motion planning capabilities but lacks integrated grasp planning for dynamic environments.
* **RightHand Robotics:** Focuses on autonomous picking and grasping in logistics but does not fully integrate motion planning.

While these companies have made significant strides in specific areas, there is a gap in the market for a solution that seamlessly integrates both grasp and motion planning in real time, especially for dynamic and cluttered environments. This is where the proposed system’s competitive edge lies. Its ability to optimize both grasping and motion simultaneously, while adapting to complex environments, offers a distinct advantage that can address the growing need for precision and flexibility in industries such as manufacturing, healthcare, and logistics.

Table 4 SWOT/TOWS Analysis for Integrated Grasp and Motion Planning Solution

|  |  |  |
| --- | --- | --- |
| **SWOT and TOWS analysis** | **Strengths**   * Superior performance in complex and dynamic environments * Real-time adaptability and precision * Unique integration of grasp and motion planning in one solution * High scalability for various industries (manufacturing, healthcare, etc.) | **Weaknesses**   * High development costs and resource requirements * Limited brand recognition in a competitive robotics market * Dependence on advanced computational resources for optimal performance |
| **Opportunities**   * Increasing demand for automation in multiple industries * Collaboration potential with academic institutions and industry leaders * Untapped market segments in healthcare, manufacturing, and logistics   Potential to lead in a niche field (integrated grasp and motion planning) | * Leverage superior technology to enter high-growth industries like healthcare and logistics where precision and adaptability are critical. * Form partnerships with leading robotics manufacturers to scale production and expand market reach.   Utilize real-time adaptability to tap into emerging markets for autonomous systems and industrial automation. | * Collaborate with research institutions to reduce development costs and gain access to technical expertise. * Build brand recognition through strategic marketing in niche sectors like precision manufacturing and robotics healthcare.   Explore external funding or venture capital to mitigate high development costs and accelerate time to market. |
| **Threats**   * Rapid technological advancements by competitors * High entry barriers due to capital requirements * Potential IP or patent challenges from competitors   Market fluctuations and changes in automation needs | * Differentiate the solution from competitors by emphasizing its unique integration of grasp and motion planning for dynamic environments. * Invest in R&D to stay ahead of technological advancements and maintain competitive edge. * Protect intellectual property through patents and partnerships to mitigate competitive risks. | * Focus on building industry partnerships to share resources and reduce capital strain. * Address scalability issues through collaborations with established players who have the infrastructure and market access needed for growth. * Implement a phased rollout to mitigate risks associated with market fluctuations or rapid technological changes. |

**SWOT Analysis:**

* Wheelen, T. L., & Hunger, J. D. (2008). *Strategic Management and Business Policy.* 11th ed. Upper Saddle River, NJ: Pearson Education.

**TOWS Analysis:**

* Weihrich, H. (1982). "The TOWS Matrix—A Tool for Situational Analysis." *Long Range Planning,* 15(2), 54-66. https://doi.org/10.1016/0024-6301(82)90120-0

## Competitive Advantage

The integrated grasp and motion planning system offers several unique selling points (USPs) that distinguish it from other solutions in the robotics marketplace. The primary advantage of the system lies in its ability to seamlessly integrate both grasp and motion planning into a unified framework. This integration allows the system to perform exceptionally well in complex, dynamic environments where conventional solutions struggle. Key USPs include:

* **Superior Performance in Complex Environments:** The system’s algorithm is designed to handle intricate and obstacle-rich environments with precision. This makes it particularly valuable in industries such as healthcare and manufacturing, where robots must navigate cluttered spaces while maintaining accuracy in their tasks.
* **Real-time Adaptability:** The system’s real-time adaptability enables it to respond dynamically to changing environments and object positions. This capability is essential for industries like logistics and autonomous robotics, where conditions can change rapidly and require immediate adjustments without human intervention.
* **Precision and Efficiency:** The solution's precision in grasping and motion is crucial for tasks that require high accuracy, such as surgical robotics or delicate assembly tasks in manufacturing. By minimizing errors in both grasping and motion, the system reduces operational risks and improves productivity.

These technical strengths provide a significant competitive advantage in the market. Many existing robotic systems treat grasp and motion planning as separate processes, leading to inefficiencies in execution. Our integrated solution, however, enhances both phases of robotic tasks, resulting in faster execution times and fewer errors. This positions the system as a leader in sectors where efficiency and adaptability are critical.

Compared to competitors like Robotiq and Universal Robots, our solution goes beyond simple motion pathfinding and grasping by offering a fully integrated system that optimizes both processes in real time. This is a key differentiator in industries where precision and adaptability are essential for success. While some competitors may excel in either grasping or motion planning, our system’s ability to do both—simultaneously and efficiently—offers a clear advantage in addressing complex, dynamic tasks.

By capitalizing on this competitive advantage, the integrated grasp and motion planning system can capture significant market share, particularly in industries that demand high-performance robotic solutions in challenging environments.

A diagram with text and symbols

Description automatically generated

Figure 4 Porter's Five forces analysis of Integrated grasp and motion planning

## Commercialization Strategy

**Bringing the Product to Market**

To successfully bring the integrated grasp and motion planning system to market, a strategic and phased approach will be employed, focusing on industries where automation and robotics are critical. The commercialization strategy will involve multiple stages, from pilot implementations to full-scale product launches, ensuring that the technology is thoroughly tested and optimized for various real-world applications.

**Target Industries**

The following industries will be the initial focus for market entry, due to their high demand for automation and precision robotics:

1. **Manufacturing:**  
   The manufacturing sector is increasingly adopting automation to enhance productivity, reduce labour costs, and improve precision in complex assembly tasks. The integrated grasp and motion planning system offers a competitive edge in tasks that require both manipulation and precision movement, making it highly suitable for assembly lines, quality control, and automated material handling.
2. **Healthcare:**  
   Robotics in healthcare, especially for tasks like surgery and rehabilitation, demands precision and reliability. The system’s ability to handle complex and dynamic environments positions it well for robotic-assisted surgeries, medical logistics, and automated diagnostics, where precision grasping and movement are critical.
3. **Logistics and Warehousing:**  
   The logistics and warehousing sector is rapidly evolving with the integration of robotics for tasks such as picking, sorting, and packing. The system’s real-time adaptability makes it an ideal solution for dynamic environments where robots need to handle various objects quickly and efficiently.

**Business Models**

Several business models will be considered for commercialization, ensuring flexibility and adaptability to different market needs:

1. **Licensing the Technology:**  
   Licensing the integrated grasp and motion planning technology to established robotics manufacturers will enable rapid scalability and widespread adoption. This model allows large-scale manufacturers to integrate the technology into their own robotic systems, providing immediate value to their customers.
2. **Offering Customized Solutions:**  
   For industries with specific needs, customized solutions will be offered. This may include tailoring the system to work with specific robotic hardware or creating specialized algorithms for unique use cases in healthcare or manufacturing. Customized solutions provide higher margins and allow for close partnerships with industry leaders.
3. **SaaS-based Consulting Services:**  
   Another potential business model is providing the system as a service (SaaS) for companies looking for consultation and ongoing support for their robotics systems. This model includes offering the software for grasp and motion planning as a subscription-based service, accompanied by consulting on how to integrate it into their workflows. This recurring revenue stream could be especially attractive in sectors like logistics, where continual support and upgrades are necessary.

## Industry Partnerships and Collaboration

Strategic partnerships with leading industry players, universities, and research institutions will be crucial for accelerating the system’s development and market penetration. Collaborations with established robotics companies can provide the necessary resources, such as advanced hardware and market access, to scale production. Partnering with academic institutions will bring cutting-edge research and technical expertise to enhance the system’s capabilities, while industry partners can help test the solution in real-world applications, refining it for commercial use.

Potential partners include robotics manufacturers for production scaling, and research institutions specializing in AI and robotics for technical development and validation. These collaborations will enable faster market entry and long-term growth in competitive industries.

**Conclusion**

The business strategy for the integrated grasp and motion planning system is built on a foundation of technical excellence, strategic partnerships, and competitive positioning. By leveraging the system's superior performance, real-time adaptability, and precision, we can differentiate it in key markets like manufacturing, healthcare, and logistics. Strategic collaborations with industry leaders and research institutions will provide the necessary resources, expertise, and market access to scale production and refine the solution for real-world applications.

The commercialization roadmap, which includes pilot implementations, full-scale development, and global expansion, ensures a structured approach to market entry and growth. With a clear focus on continuous innovation and strategic partnerships, the system is poised for long-term success and growth in the competitive robotics industry.

# Chapter 7: Conclusion

#### Summary of Research

This research focused on evaluating the RRT\* algorithm within the context of integrated grasp and motion planning, aiming to compare its performance against other algorithms like JPlusRRT and IK-RRT. The project started with implementing RRT in simple PyBullet environments and progressed to incorporating the RRT\* algorithm. Additionally, the Jogramop framework from Rudorfer et al. was employed to compare the performance of the RRT\* algorithm against other benchmarks in 20 different scenarios. Although time constraints limited the implementation of multiple algorithms in the new environment, the focus on RRT\* allowed for an in-depth assessment of its strengths and weaknesses.

#### Key Findings

The study found that RRT\* excels in less cluttered environments, showcasing its ability to refine paths iteratively for optimal results. Its asymptotic optimality makes it a strong candidate for tasks that demand high precision and efficient motion planning. However, in more complex and dense environments, the algorithm's computational demands rise due to the frequent rewiring required to optimize paths. Despite these challenges, RRT\* demonstrated robustness in dynamic and constrained environments, producing feasible and efficient paths in most scenarios.

#### Contributions to the Field

This dissertation contributes valuable insights into the application of RRT\* for integrated grasp and motion planning. By comparing RRT\* with JPlusRRT and IK-RRT in a benchmark environment, it provides a nuanced understanding of the relative strengths and weaknesses of the algorithm. The successful integration of RRT\* within the Jogramop framework adds to the body of knowledge on motion planning in constrained spaces, a critical area for real-world robotic applications. The data collected offers a solid foundation for future studies looking to refine or adapt these algorithms for more complex tasks.

#### Practical Applications

The findings from this research hold practical implications across several industries. In manufacturing, robots can benefit from the efficient motion planning offered by RRT\*, improving both precision and operational safety. In healthcare, particularly for surgical robots or assistive devices, the ability to navigate dynamic environments with high accuracy is invaluable. Autonomous systems, such as self-driving cars or service robots, can also leverage the strengths of RRT\* for efficient navigation and task execution. Although this research was conducted in simulated environments, the insights gained are readily applicable to real-world challenges faced by robotic systems.

#### Final Thoughts

Overall, this research provided a comprehensive exploration of the RRT\* algorithm’s capabilities and limitations in integrated grasp and motion planning. The study overcame challenges, such as the complexity of setting up environments and limited time for testing all algorithms in new scenarios, to deliver meaningful results. Future research can build upon this work by expanding evaluations to other algorithms, exploring hybrid approaches, and enhancing current algorithms’ efficiency by applying more creative techniques. By addressing these areas, subsequent studies can further refine integrated grasp and motion planning strategies, making them even more efficient and practical for real-world robotic systems.

# References

Abdi, A. and Park, J.H. (2023) “A Hybrid AI-Based Adaptive Path Planning for Intelligent Robot Arms,” IEEE Access, 11. Available at: https://doi.org/10.1109/ACCESS.2023.3338566.

Akinola, I. et al. (2021) “Dynamic Grasping with Reachability and Motion Awareness,” in IEEE International Conference on Intelligent Robots and Systems. Available at: https://doi.org/10.1109/IROS51168.2021.9636057.

Ali, A. and Lee, J.Y. (2020) “Integrated motion planning for assembly task with part manipulation using re-grasping,” Applied Sciences (Switzerland), 10(3). Available at: https://doi.org/10.3390/app10030749.

van den Berg, J. et al. (2010) “Centralized path planning for multiple robots: Optimal decoupling into sequential plans,” in Robotics: Science and Systems. Available at: https://doi.org/10.7551/mitpress/8727.003.0019.

Bertoni, L. et al. (2021) “Towards a Generic Grasp Planning Pipeline using End-Effector Specific Primitive Grasping Actions,” in 2021 20th International Conference on Advanced Robotics, ICAR 2021. Available at: https://doi.org/10.1109/ICAR53236.2021.9659402.

Bicchi, A. (1995) “On the Closure Properties of Robotic Grasping,” The International Journal of Robotics Research, 14(4). Available at: https://doi.org/10.1177/027836499501400402.

Bohg, J. et al. (2014) “Data-driven grasp synthesis-A survey,” IEEE Transactions on Robotics, 30(2). Available at: https://doi.org/10.1109/TRO.2013.2289018.

Calandra, R. et al. (2018) “More than a feeling: Learning to grasp and regrasp using vision and touch,” IEEE Robotics and Automation Letters, 3(4). Available at: https://doi.org/10.1109/LRA.2018.2852779.

Chauhan, R.J. and Ben-Tzvi, P. (2019) “A series elastic actuator design and control in a linkage based hand exoskeleton,” in ASME 2019 Dynamic Systems and Control Conference, DSCC 2019. Available at: https://doi.org/10.1115/DSCC2019-8996.

Ciocarlie, M.T. and Allen, P.K. (2009) “Hand posture subspaces for dexterous robotic grasping,” International Journal of Robotics Research, 28(7). Available at: https://doi.org/10.1177/0278364909105606.

Dalibard, S. and Laumond, J.P. (2010) “Control of probabilistic diffusion in motion planning,” in Springer Tracts in Advanced Robotics. Available at: https://doi.org/10.1007/978-3-642-00312-7\_29.

Dang-Vu, B.A., Porges, O. and Roa, M.A. (2016) “Interpreting manipulation actions: From language to execution,” in Advances in Intelligent Systems and Computing. Available at: https://doi.org/10.1007/978-3-319-27146-0\_14.

Dharbaneshwer, S.J. et al. (2021) “A finite element based simulation framework for robotic grasp analysis,” Proceedings of the Institution of Mechanical Engineers, Part C: Journal of Mechanical Engineering Science, 235(13). Available at: https://doi.org/10.1177/0954406220951596.

Driess, D., Ha, J.S. and Toussaint, M. (2020) “Deep Visual Reasoning: Learning to Predict Action Sequences for Task and Motion Planning from an Initial Scene Image,” in Robotics: Science and Systems. Available at: https://doi.org/10.15607/RSS.2020.XVI.003.

Elbanhawi, M. and Simic, M. (2014) “Sampling-based robot motion planning: A review,” IEEE Access. Available at: https://doi.org/10.1109/ACCESS.2014.2302442.

Fan, Z. (2023) “Multi-Point path planning for robots based on deep reinforcement learning,” in Journal of Physics: Conference Series. Available at: https://doi.org/10.1088/1742-6596/2580/1/012048.

Ferrari, C. and Canny, J. (1992) “Planning optimal grasps,” in Proceedings - IEEE International Conference on Robotics and Automation. Available at: https://doi.org/10.1109/robot.1992.219918.

Gammell, J.D., Srinivasa, S.S. and Barfoot, T.D. (2015) “Batch Informed Trees (BIT∗): Sampling-based optimal planning via the heuristically guided search of implicit random geometric graphs,” in Proceedings - IEEE International Conference on Robotics and Automation. Available at: https://doi.org/10.1109/ICRA.2015.7139620.

Garrett, C.R. et al. (2021) “Integrated Task and Motion Planning,” Annual Review of Control, Robotics, and Autonomous Systems. Available at: https://doi.org/10.1146/annurev-control-091420-084139.

Hart, P.E., Nilsson, N.J. and Raphael, B. (1968) “A Formal Basis for the Heuristic Determination of Minimum Cost Paths,” IEEE Transactions on Systems Science and Cybernetics, 4(2). Available at: https://doi.org/10.1109/TSSC.1968.300136.

Holtzen, S. et al. (2016) “Inferring human intent from video by sampling hierarchical plans,” in IEEE International Conference on Intelligent Robots and Systems. Available at: https://doi.org/10.1109/IROS.2016.7759242.

Howie Choset, Kevin Lynch, S.H. (2005) Principles of Robot Motion： Theory, Algorithms, and Implementation, The Canadian nurse.

Huang, X. et al. (2022) “Real-time grasping strategies using event camera,” Journal of Intelligent Manufacturing, 33(2). Available at: https://doi.org/10.1007/s10845-021-01887-9.

Ichnowski, J., Avigal, Y., et al. (2020) “Deep learning can accelerate grasp-optimized motion planning,” Science Robotics, 5(48). Available at: https://doi.org/10.1126/scirobotics.abd7710.

Ichnowski, J., Danielczuk, M., et al. (2020) “GOMP: Grasp-Optimized Motion Planning for Bin Picking,” in Proceedings - IEEE International Conference on Robotics and Automation. Available at: https://doi.org/10.1109/ICRA40945.2020.9197548.

Islam, F. et al. (2021) “Provably constant-time planning and replanning for real-time grasping objects off a conveyor belt,” International Journal of Robotics Research, 40(12–14). Available at: https://doi.org/10.1177/02783649211027194.

Kahn, G. et al. (2018) “Self-Supervised Deep Reinforcement Learning with Generalized Computation Graphs for Robot Navigation,” in Proceedings - IEEE International Conference on Robotics and Automation. Available at: https://doi.org/10.1109/ICRA.2018.8460655.

Karaman, S. et al. (2011) “Anytime motion planning using the RRT,” in Proceedings - IEEE International Conference on Robotics and Automation. Available at: https://doi.org/10.1109/ICRA.2011.5980479.

Karaman, S. and Frazzoli, E. (2011) “Incremental sampling-based algorithms for optimal motion planning,” in Robotics: Science and Systems. Available at: https://doi.org/10.15607/rss.2010.vi.034.

Kavraki, L.E. et al. (1996) “Probabilistic roadmaps for path planning in high-dimensional configuration spaces,” IEEE Transactions on Robotics and Automation, 12(4). Available at: https://doi.org/10.1109/70.508439.

Kohlbrecher, S. et al. (2014) “Hector open source modules for autonomous mapping and navigation with rescue robots,” in Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics). Available at: https://doi.org/10.1007/978-3-662-44468-9\_58.

Kopicki, M. et al. (2011) “Learning to predict how rigid objects behave under simple manipulation,” in Proceedings - IEEE International Conference on Robotics and Automation. Available at: https://doi.org/10.1109/ICRA.2011.5980295.

Lavalle, S.M. (2006) PLANNING ALGORITHMS. Available at: https://msl.cs.uiuc.edu/planning/bookbig.pdf (Accessed: August 4, 2024).

Lee, H.C., Yaniss, T. and Lee, B.H. (2012) “Grafting: A path replanning technique for rapidly-exploring random trees in dynamic environments,” Advanced Robotics, 26(18). Available at: https://doi.org/10.1080/01691864.2012.703301.

Leu, J. et al. (2022) “Robust Task Planning for Assembly Lines with Human-Robot Collaboration.” Available at: http://arxiv.org/abs/2204.07936.

Li, G. et al. (2023) “Design of Digital Planner and 3D Vision System for Robot Bin Picking,” in 2023 8th IEEE International Conference on Advanced Robotics and Mechatronics, ICARM 2023. Available at: https://doi.org/10.1109/ICARM58088.2023.10218895.

Li, T.-Y. and Huang, P.-Z. (2007) “Planning Versatile Motions for Humanoid in a Complex Environment,” in Humanoid Robots: New Developments. Available at: https://doi.org/10.5772/4888.

Liniger, A. and van Gool, L. (2020) “Safe Motion Planning for Autonomous Driving using an Adversarial Road Model,” in Robotics: Science and Systems. Available at: https://doi.org/10.15607/RSS.2020.XVI.044.

Mahler, J. et al. (2017) “Dex-Net 2.0: Deep learning to plan Robust grasps with synthetic point clouds and analytic grasp metrics,” in Robotics: Science and Systems. Available at: https://doi.org/10.15607/rss.2017.xiii.058.

Mahler, J. et al. (2019) “Learning ambidextrous robot grasping policies,” Science Robotics, 4(26). Available at: https://doi.org/10.1126/scirobotics.aau4984.

Manzinger, S., Pek, C. and Althoff, M. (2021) “Using Reachable Sets for Trajectory Planning of Automated Vehicles,” IEEE Transactions on Intelligent Vehicles, 6(2). Available at: https://doi.org/10.1109/TIV.2020.3017342.

Mishra, B., Schwartz, J.T. and Sharir, M. (1987) “On the existence and synthesis of multifinger positive grips,” Algorithmica, 2(1–4). Available at: https://doi.org/10.1007/BF01840373.

Muhayyuddin, Akbari, A. and Rosell, J. (2015) “Ontological physics-based motion planning for manipulation,” in IEEE International Conference on Emerging Technologies and Factory Automation, ETFA. Available at: https://doi.org/10.1109/ETFA.2015.7301404.

Osa, T., Peters, J. and Neumann, G. (2018) “Hierarchical reinforcement learning of multiple grasping strategies with human instructions,” Advanced Robotics, 32(18). Available at: https://doi.org/10.1080/01691864.2018.1509018.

Paxton, C. et al. (2017) “Combining neural networks and tree search for task and motion planning in challenging environments,” in IEEE International Conference on Intelligent Robots and Systems. Available at: https://doi.org/10.1109/IROS.2017.8206505.

Perez, A. et al. (2012) “LQR-RRT\*: Optimal sampling-based motion planning with automatically derived extension heuristics,” in Proceedings - IEEE International Conference on Robotics and Automation. Available at: https://doi.org/10.1109/ICRA.2012.6225177.

Prats, M., Sanz, P.J. and del Pobil, A.P. (2007) “Task-oriented grasping using hand preshapes and task frames,” in Proceedings - IEEE International Conference on Robotics and Automation. Available at: https://doi.org/10.1109/ROBOT.2007.363582.

Qureshi, A.H. et al. (2019) “Motion planning networks,” in Proceedings - IEEE International Conference on Robotics and Automation. Available at: https://doi.org/10.1109/ICRA.2019.8793889.

Qureshi, A.H. and Ayaz, Y. (2015) “Intelligent bidirectional rapidly-exploring random trees for optimal motion planning in complex cluttered environments,” Robotics and Autonomous Systems, 68. Available at: https://doi.org/10.1016/j.robot.2015.02.007.

Rosell, J. et al. (2019) “Planning grasping motions for humanoid robots,” in International Journal of Humanoid Robotics. Available at: https://doi.org/10.1142/S0219843619500415.

Rubert, C. et al. (2018) “Characterisation of Grasp Quality Metrics,” Journal of Intelligent and Robotic Systems: Theory and Applications, 89(3–4). Available at: https://doi.org/10.1007/s10846-017-0562-1.

Rudorfer, M., Hartvich, J. and Vonásek, V. (no date) A Framework for Joint Grasp and Motion Planning in Confined Spaces.

Sebastiani, E. et al. (2017) “Dealing with on-line human-robot negotiations in hierarchical agent-based task planner,” in Proceedings International Conference on Automated Planning and Scheduling, ICAPS. Available at: https://doi.org/10.1609/icaps.v27i1.13862.

Tahir, Z. et al. (2018) “Potentially guided bidirectionalized RRT\* for fast optimal path planning in cluttered environments,” Robotics and Autonomous Systems, 108. Available at: https://doi.org/10.1016/j.robot.2018.06.013.

Tai, L., Li, S. and Liu, M. (2016) “A deep-network solution towards model-less obstacle avoidance,” in IEEE International Conference on Intelligent Robots and Systems. Available at: https://doi.org/10.1109/IROS.2016.7759428.

Tsuji, T. et al. (2010) “Grasp Planning for a Multifingered Hand with a Humanoid Robot,” Journal of Robotics and Mechatronics, 22(2). Available at: https://doi.org/10.20965/jrm.2010.p0230.

Vahrenkamp, N. et al. (2010) “Integrated grasp and motion planning,” in Proceedings - IEEE International Conference on Robotics and Automation. Available at: https://doi.org/10.1109/ROBOT.2010.5509377.

Wan, W. and Harada, K. (2016) “Integrated assembly and motion planning using regrasp graphs,” Robotics and Biomimetics, 3(1). Available at: https://doi.org/10.1186/s40638-016-0050-2.

Wan, W. and Harada, K. (2017) “Regrasp planning using 10,000s of grasps,” in IEEE International Conference on Intelligent Robots and Systems. Available at: https://doi.org/10.1109/IROS.2017.8206011.

Wang, L., Xiang, Y. and Fox, D. (2020) “Manipulation Trajectory Optimization with Online Grasp Synthesis and Selection,” in Robotics: Science and Systems. Available at: https://doi.org/10.15607/RSS.2020.XVI.033.

Xin, P. et al. (2023) “Improved Bidirectional RRT\* Algorithm for Robot Path Planning,” Sensors, 23(2). Available at: https://doi.org/10.3390/s23021041.

Yan, W. et al. (2019) “Precision Grasp Planning for Multi-Finger Hand to Grasp Unknown Objects,” Robotica, 37(8). Available at: https://doi.org/10.1017/S0263574719000031.

Yang, Y. et al. (2018) “HDRM: A Resolution Complete Dynamic Roadmap for Real-Time Motion Planning in Complex Scenes,” IEEE Robotics and Automation Letters, 3(1). Available at: https://doi.org/10.1109/LRA.2017.2773669.

Yi, J. et al. (2022) “Path planning of a manipulator based on an improved P\_RRT\* algorithm,” Complex and Intelligent Systems, 8(3). Available at: https://doi.org/10.1007/s40747-021-00628-y.

Yu, J. and LaValle, S.M. (2016) “Optimal Multirobot Path Planning on Graphs: Complete Algorithms and Effective Heuristics,” IEEE Transactions on Robotics, 32(5). Available at: https://doi.org/10.1109/TRO.2016.2593448.

Yu, J., Su, Y. and Liao, Y. (2020) “The Path Planning of Mobile Robot by Neural Networks and Hierarchical Reinforcement Learning,” Frontiers in Neurorobotics, 14. Available at: https://doi.org/10.3389/fnbot.2020.00063.

Zafra-Urrea, R.M., López-Damian, E. and Santana-Díaz, A. (2023) “Grasp Planning Based on Metrics for Collaborative Tasks Using Optimization,” Applied Sciences (Switzerland), 13(17). Available at: https://doi.org/10.3390/app13179603.

Zhang, H. and Zhu, Z. (2020) “Sampling-based motion planning for free-floating space robot without inverse kinematics,” Applied Sciences (Switzerland), 10(24). Available at: https://doi.org/10.3390/app10249137.

Ziegler, J. et al. (2014) “Making bertha drive-an autonomous journey on a historic route,” IEEE Intelligent Transportation Systems Magazine, 6(2). Available at: https://doi.org/10.1109/MITS.2014.2306552.

Zucker, M. et al. (2013) “CHOMP: Covariant Hamiltonian optimization for motion planning,” International Journal of Robotics Research, 32(9–10). Available at: https://doi.org/10.1177/0278364913488805.

# Appendix

**Algorithm:** RRT\* Algorithm for sampling-based motion planning, adapted from Karaman et al.'s paper *Anytime Motion Planning using the RRT*\* (Karaman & Frazzoli, 2011).

|  |
| --- |
| **Algorithm 1:** T = (V, E) ← RRT\*(zinit) |
| 1: T ← InitializeTree();  2: T ← InsertNode(∅, zinit, T );  3: for i = 1 to i = N do  4: zrand ← Sample(i);  5: znearest ← Nearest(T , zrand);  6: (xnew, unew, Tnew) ← Steer(znearest, zrand);  7: if ObstacleFree(xnew) then  8: Znear ← Near(T , znew, |V|);  9: zmin ← ChooseParent(Znear, znearest, znew, xnew);  10: T ← InsertNode(zmin, znew, T );  11: T ← ReWire(T , Znear, zmin, znew);  12: return T |

|  |
| --- |
| **Algorithm 2:** zmin ← ChooseParent(Znear, znearest, xnew) |
| 1: zmin ← znearest;  2: cmin ← Cost(znearest) + c(xnew);  3: for znear ∈ Znear do  4: (x', u', T') ← Steer(znear, znew);  5: if ObstacleFree(x') and x'(T') = znew then  6: c' = Cost(znear) + c(x');  7: if c' < Cost(znew) and c' < cmin then  8: zmin ← znear;  9: cmin ← c';  10: return zmin |

|  |
| --- |
| **Algorithm 3:** T ← ReWire(T , Znear, zmin, znew) |
| 1: for znear ∈ Znear \ {zmin} do  2: (x', u', T') ← Steer(znew, znear);  3: if ObstacleFree(x') and x'(T') = znear and Cost(znew) + c(x') < Cost(znear) then  4: T ← ReConnect(znew, znear, T);  5: return T |