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

Declaration:

I declare that I have personally prepared this assignment. The work is my own, carried out personally by me unless otherwise stated and has not been generated using paid for assessment writing services or Artificial Intelligence tools unless specified as a clearly stated approved component of the assessment brief. All sources of information, including quotations, are acknowledged by means of the appropriate citations and references. I declare that this work has not gained credit previously for another module at this or another University, save for permitted elements which formed part of an associated proposal linked directly to this submission.

I understand that plagiarism, collusion, copying another student and commissioning (which for the avoidance of doubt includes the use of essay mills and other paid for assessment writing services, as well as unattributed use of work generated by Artificial Intelligence tools) are regarded as offences against the University's Assessment Regulations and may result in formal disciplinary proceedings.

I understand that by submitting this assessment, I declare myself fit to be able to undertake the assessment and accept the outcome of the assessment as valid.

Student signature: Salah Ebrahimpour

Date: 09/29/2024

Abstract

This dissertation focuses on the implementation and evaluation of several motion-planning algorithms—specifically RRT*, JPlusRRT, IK-RRT, and BIK-RRT—within the context of integrated grasp and motion planning tasks. The research explores how these algorithms perform across key metrics, such as planning time, success rate, and path length, in both the PyBullet and Jogramop (Rudorfer et al., n.d.) environments. These environments feature multiple predefined scenarios that simulate complex robotic tasks, such as reaching goals located on or under a table, to assess the computational efficiency and effectiveness of each algorithm.

The primary objective was to compare how well each algorithm balances the trade-off between computational speed and path optimization, while also evaluating their adaptability to different environmental configurations, including varying obstacle layouts and complexities. RRT* excelled in path quality but incurred higher computational costs, making it suitable for tasks requiring high precision. In contrast, JPlusRRT offered faster exploration and adaptability, making it ideal for dynamic environments. IK-RRT and BIK-RRT were more focused on kinematic precision but showed slower performance in certain complex scenarios.

This research contributes valuable insights into the practical application of these algorithms in real-world robotics, and highlights which algorithms are better suited for specific tasks. The findings hold implications for industries such as manufacturing, healthcare, and autonomous systems, where robotic motion planning must balance speed, accuracy, and adaptability. Ultimately, the study emphasizes that no single algorithm provides an all-encompassing solution, and the choice of algorithm should be informed by the specific requirements of the task and environment.

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

Figures

Tables

Abbreviations

1. RRT: Rapidly-exploring Random Tree
2. RRT*: RRT-Star (an optimal variant of RRT)
3. JPlusRRT: Jacobian Pseudoinverse-based RRT
4. IK-RRT: Inverse Kinematics RRT
5. BIK-RRT: Bidirectional Inverse Kinematics RRT
6. PRM: Probabilistic Roadmap
7. IK: Inverse Kinematics
8. DOF: Degrees of Freedom
9. MPC: Model Predictive Control
10. C-space: Configuration Space
11. C_{free}: Free Configuration Space
12. Cobs: Obstacle Configuration Space
13. J+: Jacobian Pseudoinverse

Chapter 1: Introduction

1.1. Project background

As robotics technology advances and the need for intelligent robotic systems grows, the integration of sophisticated tools for autonomous manipulation tasks becomes increasingly vital. The growing demand for automation and intelligent systems has led to a focus on enhancing the capabilities of robotic systems, particularly in intelligent robotic arms. These robotic arms, designed to mimic the movements and functionality of human arms, play a crucial role in various industries. As depicted in Fig. 1, these arms are engineered for precise manipulation tasks, making them essential for modern automation and intelligent systems (Yi et al., 2022).

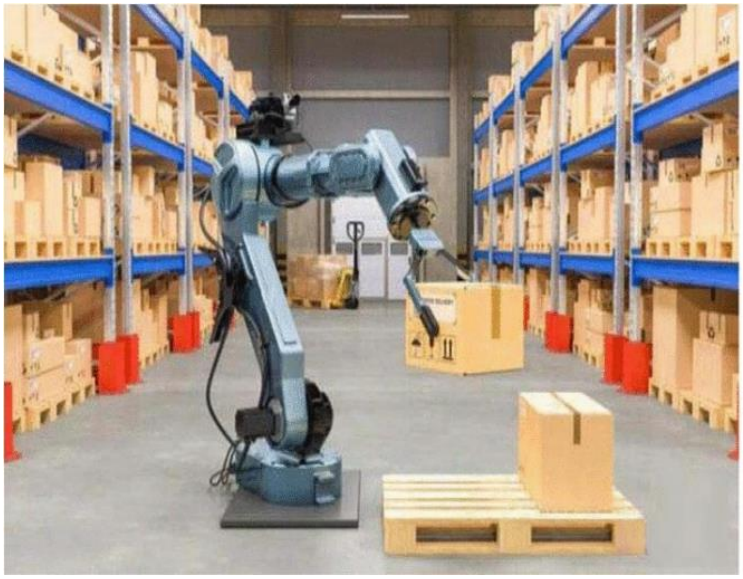

Figure 1 A robotic hand picking up goods in warehousing industry (Yi et al., 2021).

Traditionally, robotic systems have treated grasp and motion planning as separate processes. Grasp planning focuses on determining how a robot should grip an object, considering factors such as its shape, weight, and texture. In contrast, motion planning addresses the path the robot must follow and the obstacles it must navigate to complete tasks (Muhayyuddin et al., 2015). However, separating these processes can result in suboptimal performance, especially in complex environments where the configuration of the grasp can directly impact the feasibility of the motion (Ali and Lee, 2020). 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.

1.2. Research Objectives

The primary objective of this research is to implement and evaluate several motion-planning algorithms—specifically RRT, RRT, JPlusRRT, 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, including planning time, success rate, and path length. By implementing these algorithms across multiple predefined scenarios, the research seeks to evaluate their performance not only in terms of computational efficiency and the quality of generated paths but also in identifying which algorithm performs best in specific practical situations. The study focuses on understanding how each algorithm balances speed and path optimization to offer insights into their suitability for various robotic motion planning tasks. Additionally, the comparison highlights how each algorithm adapts to different environmental configurations, such as changes in obstacle density and layout complexity, which provides insights into selecting the most suitable algorithm for various real-world applications.

1.3. Research Questions/Hypotheses

- How do the motion-planning algorithms—RRT, RRT*, JPlusRRT, IK-RRT, and BIK-RRT—perform in terms of planning time, success rate, and path length across different predefined scenarios?
- How to integrate motion planning algorithms with grasp planning? And how grasp planning affects the overall efficiency of the algorithms?
- What are the strengths and limitations of each algorithm in optimizing computational efficiency while maintaining path quality in integrated grasp and motion planning tasks?
- Which practical scenarios or tasks are best suited to each algorithm, based on their individual performance characteristics?

The research questions aim to assess the performance of various motion-planning algorithms—RRT*, JPlusRRT, IK-RRT, and BIK-RRT—based on key metrics like planning time, success rate, and path length. Additionally, the study seeks to understand the strengths and limitations of each algorithm, particularly how they balance computational efficiency with path quality. Another key

objective is to identify which practical situations or tasks each algorithm is most suitable for, depending on its performance in different environments. Lastly, the research explores how effectively these algorithms adapt to varying environmental conditions, such as obstacle diversity and layout complexity, and how this impacts their practical application in real-world robotic systems.

1.4. Significance of the Study

This research focuses on evaluating the performance of several motion-planning algorithms—RRT*, JPlusRRT, 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 path length, the study provides critical insights into the strengths and limitations of these algorithms when applied in real-world scenarios which require 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., n.d.) . 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 by offering a more reliable understanding of how these algorithms operate in environments with complex constraints.

1.5. Scope and Limitations

The scope of this study involves implementing and comparing integrated grasp and motion planning algorithms in simulated environments, such as PyBullet, and using benchmark scenarios provided by Jogramop framework (Rudorfer et al., n.d.). 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 length.

However, this research was constrained by limited time, which affected the extent to which algorithms could be fine-tuned across various environments. The fine-tuning of parameters was tailored to the specific scenarios tested in the simulation, raising concerns about how well these results generalize to other environments. Additionally, while simulations offer a controlled space for testing and refining algorithms, they fail to fully replicate the complexities of real-world applications, such as sensor inaccuracies, hardware limitations, or unexpected environmental factors. These limitations suggest the need for future work to validate the algorithms in physical environments and explore more generalized parameter settings to ensure their broader applicability in practical robotic systems.

1.6. Thesis Structure

The thesis is structured to provide a comprehensive exploration of the implementation and evaluation of motion planning algorithms within integrated grasp and motion planning tasks.

Chapter 2 offers a detailed literature review, setting the context for integrated grasp and motion planning. It explores grasp and motion planning algorithms. This chapter also identifies gaps in the existing literature and summarizes the key insights.

Chapter 3 presents the problem description, laying the groundwork for the research objectives and outlining the specific challenges addressed in the study.

Chapter 4 focuses on the methodology, detailing the implementation of the algorithms, the research design, and the experimental setup. It describes the metrics used for evaluation and provides an overview of the results from both the PyBullet and Jogramop environments. The chapter concludes with a discussion of the challenges encountered during the study and the adaptations made to address them.

Chapter 5 provides an evaluation and reflection on the project's achievements, comparing the outcomes against the original objectives. It discusses the evidence gathered from performance studies, compares the chosen algorithms with alternatives, and outlines areas for improvement and future research.

Chapter 6 shifts the focus to project management, detailing how the research was organized, the timelines followed, and the key milestones achieved throughout the study.

Chapter 7 discusses the business strategy and market integration of the research. It explores the growth potential of the robotics marketplace, conducts a competitive analysis, and proposes strategies for commercialization and industry partnerships.

Finally, Chapter 8 concludes the thesis by summarizing the key findings and contributions of the research, highlighting its practical implications and potential for future exploration.

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 and Simic, 2014), the simultaneous planning of both grasping an object and coordinating the robot's motion remains a notable challenge (Wang et al., 2020). 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).

Traditionally, grasp planning and arm motion planning have been treated as distinct tasks, which can lead to limitations in grasp options and longer computational time (Fontanals et al., 2015). 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) as depicted in Fig 2. 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.

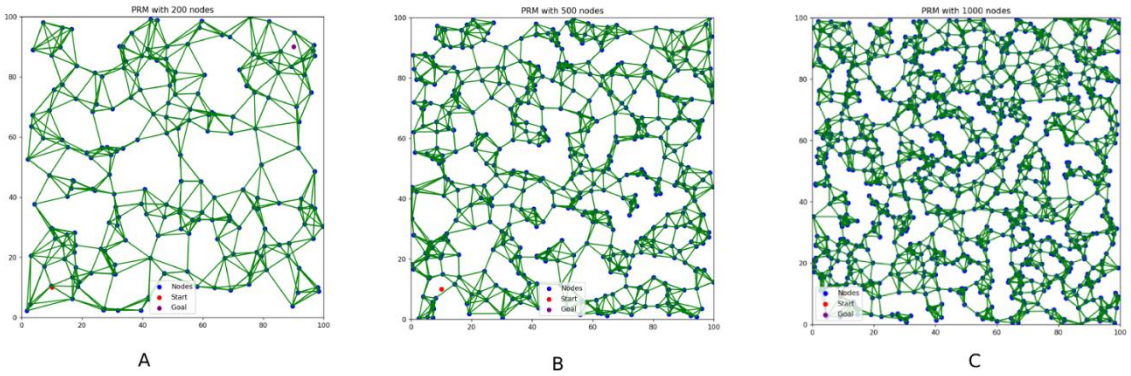

Figure 2 PRM algorithm exploring the configuration space with varying node counts: A) 200 nodes, B) 500 nodes, C) 1000 nodes.

This led to an investigation of the Rapidly Exploring Random Tree (RRT) algorithm, known for its ability to efficiently explore high-dimensional spaces without requiring pre-constructed paths.

The Rapidly-exploring Random Tree (RRT) algorithm was first introduced by Steven M. LaValle in 1998. As shown in Figure 3, 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.

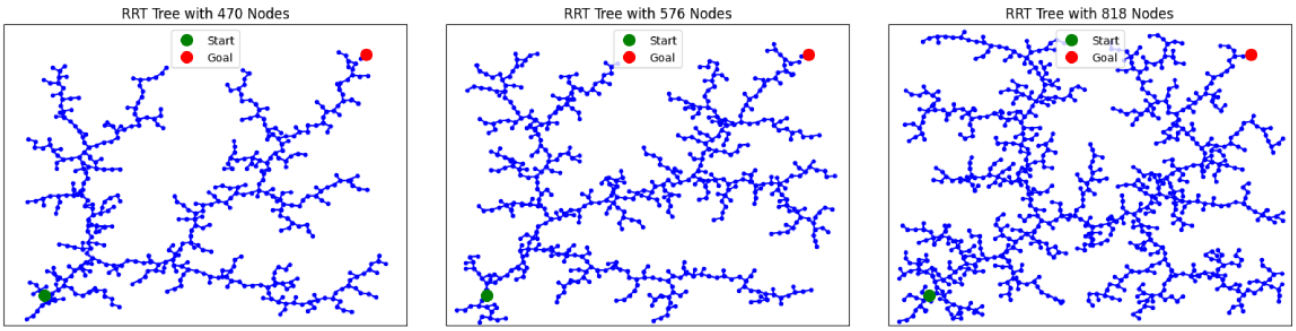

Figure 3 RRT algorithm exploring the configuration space. Max iteration= 1000.

Further investigation into RRT's extensions revealed promising advancements. Algorithms like RRT*, JPlusRRT, 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* (Karaman and Frazzoli, 2011) 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 C represents all possible positions and orientations a robot can occupy. A specific configuration $q \in C$ is defined by a set of variables, such as the robot's joint angles or positions in the workspace. The free configuration space $C_{free} \subseteq C$ consists of all configurations where the robot does not collide with obstacles, while the obstacle space $C_{obs} \subseteq C$ represents configurations where collisions occur.

Motion planning algorithms, including the ones discussed in this section, operate within C_{free} searching for a collision-free path from a start configuration q_{init} to a goal configuration q_{goal}. The complexity of navigating C_{free}, 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 X, is the core of all sampling-based methods. The procedure, denoted as $\mathrm{Sample}(X)$, generates independent and identically distributed points from X. An extension of this, $\mathrm{SampleFree}(X)$, ensures that samples are drawn from the free space X_{free}, while avoiding obstacles.

Nearest Neighbor: Given a graph $G = (V, E)$ where $V \subset X$, the nearest neighbor function, $\mathrm{Nearest}(G, x)$, returns the vertex $v \in V$ that is closest to the point x, based on a distance metric, often Euclidean.

Steering: This function, $\mathrm{Steer}(x, y)$, generates a point that moves closer to the target y from the point x, while maintaining a maximum allowable step size, ensuring gradual and feasible transitions.

Collision Test: The function $\mathrm{CollisionFree}(x, x')$ evaluates whether the straight-line path between two points lies entirely within X_{free}.

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, X_{free}, 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 by 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:

Algorithm* 1: *PRM
1 $V \leftarrow \emptyset$; $E \leftarrow \emptyset$;
2 ***for*** $i = 0, \ldots, n$ ***do***
3 $x_{rand} \leftarrow SampleFree_i$
4 $U \leftarrow Near(G = (V, E), x_{rand}, r)$;
5 $V \leftarrow V \cup x_{rand}$;

6 $\boldsymbol{foreach}\ u \in U, in\ order\ of\ increasing\ \left||u - x_{rand}|\right|, \boldsymbol{do}$

7 $\quad \boldsymbol{if}\ x_{rand}\ \boldsymbol{and}\ u\ are\ not\ in\ the\ same\ connected\ component\ of\ G = (V, E)\ then$

8 $\quad\quad \boldsymbol{if}\ CollisionFree(x_{rand}, u)\ \boldsymbol{then}\ E \leftarrow E \cup (x_{rand}, u), (u, x_{rand});$

9 $\boldsymbol{return}\ G = (V, E);$

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, X_{free}, 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, as illustrated in Figure 4, where the algorithm expands by sampling a random state X_{rand} , identifying the nearest vertex X_{near}, and extending the tree toward X_{rand} to obtain X_{new}.

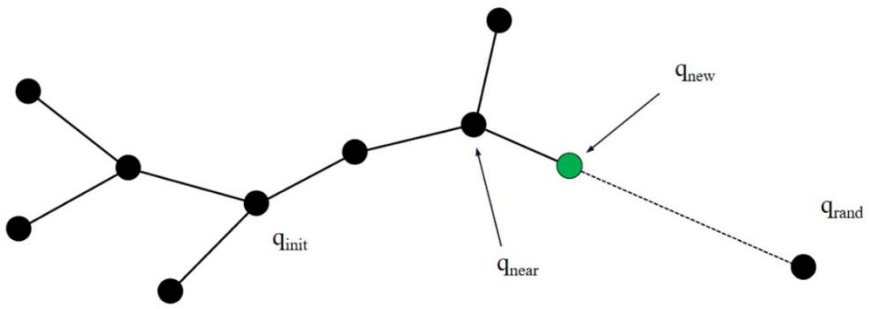

Figure 4 Schematic of RRT algorithm expansion, where the algorithm samples a random state q_{rand} , identifies the nearest vertex q_{near} , and extends the tree towards q_{rand} to obtain q_{near} (Wu, Meng, Zhao & Wu, 2021)

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 x_{init} as the sole vertex, and no edges. At each iteration, a random point $x_{rand} \in X_{free}$ is sampled. The nearest node $v \in V$ from the current tree is then identified, and an attempt is made to steer from v toward x_{rand}. If this new connection is valid (i.e., it avoids obstacles), the new point x_{new} is added to the tree as a vertex, and the edge (v, x_{new}) is added to the edge set.

This process continues until a specified number of iterations n is completed, or the tree reaches the goal region. In the original RRT algorithm, the iteration could stop as soon as a path to the goal is found.

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

1. $V \leftarrow x_{init}; E \leftarrow \emptyset;$
2. $\boldsymbol{for}\ i = 1, \ldots, n\ do$
3. $x_rand \leftarrow SampleFree_i;$
4. $x_{nearest} \leftarrow Nearest(G = (V, E), x_{rand});$
5. $x_{new} \leftarrow Steer(x_{nearest}, x_{rand});$
6. $\boldsymbol{if}\ ObstacleFree(x_{nearest}, x_{new})\ \boldsymbol{then}$
7. $V \leftarrow V \cup x_{new}; E \leftarrow E \cup (x_{nearest}, x_{new});$
8. $return\ G = (V, E);$

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 (Figure 4). 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.

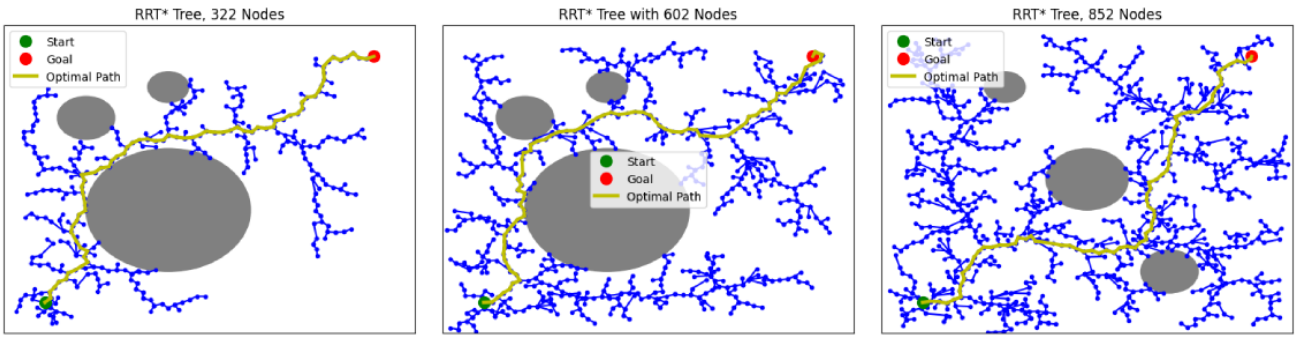

Figure 5 RRTStar explores the configuration space and finds optimal path (Yellow) by creating a tree from start position (Green) to the end position (Red) while avoiding obstacles (Grey).

In RRT*, when a new node x_{new} 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 $r(n)$ 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 (see Figure 6 for the flowchart of the RRT* algorithm's expansion process, including rewiring steps).

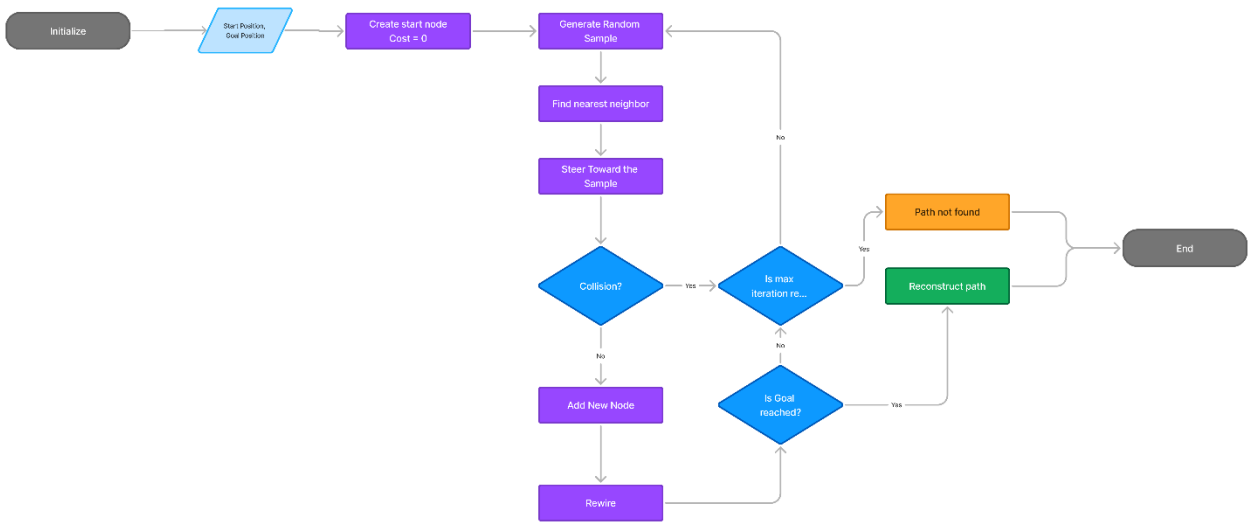

Figure 6 Flowchart of the RRT algorithm's expansion process. The diagram illustrates the sequence from initialization, random sampling, tree expansion, and rewiring to path optimization.*

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 η controls the step size, limiting how far new nodes can extend, ensuring smoother connections and more efficient paths.
2. **Rewiring for Optimality**: After adding $\mathrm{x_{new}}$, 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 c 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 $r(n)$ for connecting to nearby nodes is defined as:

$$r(n) = \gamma \left(\frac{\log n}{n}\right)^{\frac{1}{d}}$$

   where n is the number of nodes and d is the dimensionality of the space. This radius decreases with n, ensuring that connections are more localized as the tree grows.

2. **Cost Function**: For each node $v \in V$, the cost to reach that node is:

$$c(v) = c(Parent(v)) + Distance(Parent(v), v)$$

   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 x_{new}, RRT* rewires any nearby node x_{near} if the path through x_{new} reduces the overall cost:

$$c(x_{near}) > c(x_{new}) + Distance(x_{new}, x_{near})$$

If this condition holds, the edge from $Parent(x_{near})$ to x_{near} is removed, and a new edge fromx_{new} to x_{near} is added.

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

Algorithm 3: RRT*

$V \leftarrow x_{init}; E \leftarrow \emptyset;$

for $i = 1, \ldots, n$ **do**

  $x_{rand} \leftarrow SampleFree_i;$

  $x_{nearest} \leftarrow Nearest(G = (V, E), x_{rand});$

  $x_{new} \leftarrow Steer(x_{nearest}, x_{rand});$

  if $ObstacleFree(x_{nearest}, x_{new})$ **then**

    $X_{near} \leftarrow Near\left(G = (V, E), x_{new}, min\gamma_{RRT*}\left(log\big(card(V)\big)/card(V)\right)^{(1/d)}, \eta\right);$

    $V \leftarrow V \cup x_{new};$

    $x_{min} \leftarrow x_{nearest}; c_{min} \leftarrow Cost(x_{nearest}) + c\big(Line(x_{nearest}, x_{new})\big);$

    foreach $x_{near} \in X_{near}$ **do** // Rewiring

      if $CollisionFree(x_{near}, x_{new}) \wedge Cost(x_{near}) + c\big(Line(x_{near}, x_{new})\big) < c_{min}$ **then**

        $x_{min} \leftarrow x_{near}; c_{min} \leftarrow Cost(x_{near}) + c\big(Line(x_{near}, x_{new})\big)$

    $E \leftarrow E \cup (x_{min}, x_{new});$

    foreach $x_{near} \in X_{near}$ **do** // Rewire the tree

15 $\boldsymbol{if}\ CollisionFree(x_{new}, x_{near}) \wedge Cost(x_{new}) + c\big(Line(x_{new}, x_{near})\big) < Cost(x_{near})$

$\boldsymbol{then}\ x_{parent} \leftarrow Parent(x_{near});$

16 $E \leftarrow \Big(E\left(x_{parent}, x_{near}\right)\Big) \cup (x_{new}, x_{near})$

17 $\boldsymbol{return}\ G\ =\ (V, E);$

2.2.6 JPlusRRT: Enhanced Speed and Adaptability

JPlusRRT 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, JPlusRRT integrates grasp planning directly into the RRT framework to quickly find feasible paths. Instead of relying solely on inverse kinematics (IK) solvers, it uses a Jacobian pseudoinverse-based approach to guide the robot's end-effector towards grasping configurations in the task space. This method allows for direct control over the robot's arm movements, especially in complex environments where multiple grasping poses are predefined. By using this approach, JPlusRRT avoids the computational cost of searching for explicit IK solutions, which makes it more efficient in scenarios requiring rapid decision-making.

The core of JPlusRRT 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, J^{+}, 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 JPlusRRT:

1. **Task-Space Guided Exploration:** Instead of only focusing on the configuration space ($C - space$), JPlusRRT 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:** JPlusRRT incorporates a set of feasible grasps and iteratively selects a grasp as the target for the robot's end-effector.

4. **Simplified Approach:** JPlusRRT 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, J^+, is used to calculate changes in joint angles Δq based on the difference between the desired task-space position Δp and the current position:

$$\Delta q = J^+ \Delta p$$

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

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

$$q_{\text{new}} = q_{\text{near}} + J^+ \cdot \Delta p$$

  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 JPlusRRT:

$Algorithm: JPplusRRT\ (q\_start, p\_obj, grasp\_candidates)$
1. $RRT.AddConfiguration(q\_start)$ 2. $while\ (!TimeOut())\ do$ 3. $\quad ExtendRandomly(RRT)$ 4. $\quad if\ (rand() < pExtendToGoal)\ then$ 5. $\quad\quad Solution \leftarrow ExtendToGoal(RRT, p\_obj, grasp\_candidates)$ 6. $\quad\quad if\ (Solution != NULL)\ then$ 7. $\quad\quad\quad return\ PrunePath(Solution)$

8. $end\ if$
9. $end\ if$
10. $end\ while$

Function: $ExtendToGoal\ (RRT, p\_obj, grasp\_candidates)$

1. $grasp \leftarrow GetRandomGrasp(grasp\_candidates)$
2. $p\_target \leftarrow ComputeTargetPose(p\_obj, grasp)$
3. $q\_near \leftarrow GetNearestNeighbor(RRT, p\_target)$
4. $repeat$
5. $p\_near \leftarrow ForwardKinematics(q\_near)$
6. $\Delta p \leftarrow p\_target - p\_near$
7. $\Delta q \leftarrow J + (q\_near) * LimitCartesianStepSize(\Delta p)$
8. $q\_near \leftarrow q\_near + \Delta q$
9. $if\ (Collision(q\_near)\ ||\ !InJointLimits(q\_near))\ then$
10. $return\ NULL$
11. $RRT.AddConfiguration(q\_near)$
12. $until\ (Length(\Delta p) < Threshold)$
13. $return\ BuildSolutionPath(q\_near)$

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 **path length**.

- **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).
- **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 and Frazzoli, 2011).
- **Path Length**: This refers to the total distance of the generated path from the start to the goal. While RRT is efficient at exploring large spaces quickly, it often produces longer, non-optimal paths. RRT*, on the other hand, optimizes the path by continuously refining it as the tree grows, leading to shorter, more efficient routes (Karaman et al., 2011). Path length is a critical metric in environments where both speed and efficiency are required.

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 and ensures 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 different environments. These benchmarks allow researchers to test algorithms like RRT and PRM in environments with consistent obstacle density and path complexity. Additionally, frameworks such as python's **PyBullet** are often employed to simulate these environments.

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).

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 (Fan, 2023; Lavalle, 2006). 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 and 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 and Van Gool, 2020; Zhang and 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 (Kahn et al., 2018; Tai et al., 2016). Machine learning models trained on large datasets have significantly reduced planning times by predicting feasible grasps or paths based on past experiences (Calandra et al., 2018; Mahler et al., 2017). These hybrid approaches show promise in overcoming the limitations of traditional sampling-based methods, especially in environments where real-time adaptability is crucial.

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 and will influence 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 and Allen, 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.

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, which enables 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 analysing 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 and Harada, 2017). These methods provide a detailed understanding of the object’s shape which allows 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 and 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., 2011). These advancements help robots dynamically adjust their grasps in response to changes, ensuring robustness in execution.

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. 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 and Allen, 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., 2016). By learning from previous tasks, robots can generalize their grasp and motion strategies across different objects and environments and enhance 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 and 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, this research focuses on two critical areas: scalability and generalization to novel tasks. By implementing and evaluating multiple motion-planning algorithms—RRT, RRT*, JPlusRRT, IK-RRT, and BIK-RRT—across increasingly complex environments, we aim to understand how well these methods scale when applied to high-dimensional tasks. Additionally, we investigate the generalization of these algorithms to unstructured and novel scenarios. This approach not only contributes to improving the adaptability of motion-planning algorithms but also ensures that they can be applied effectively in real-world applications that demand both scalability and flexibility.

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 and Allen, 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 and 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 and 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.

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. To help test algorithms in near real-world scenarios, this research tests the robustness of the algorithms in various simulated 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 length of the paths generated by each algorithm which means the distance that needs to be traveled using the path from the star position to the end position.
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 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 different environments. The goal is to evaluate several algorithms, 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 and evaluation of several motion planning algorithms—RRT, RRT*, JPlusRRT, IK-RRT, and BIK-RRT—using Python. These algorithms were selected to assess their performance in robotic motion planning tasks, particularly in environments requiring precise navigation around obstacles.

The experimental process is divided into two phases. The first phase focuses on implementing and testing the algorithms in the PyBullet environment, where key performance metrics such as path length, planning time, and success rate were evaluated in PyBullet scenarios. In the second phase, the RRT* algorithm was implemented within the Jogramop framework, which provides standardized benchmark scenarios designed to evaluate the algorithm's performance in more complex, obstacle-rich environments.

This chapter will cover the research design, the rationale for the selection of algorithms and frameworks, the experimental setup, the evaluation metrics, and the presentation of results. The goal is to demonstrate the practical implementation of these algorithms and to compare their effectiveness in solving motion planning problems in various robotic contexts.

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 G of potential grasps, where each grasp $g = (T_5, S_g)$ consists of a grasp pose $\mathrm{T_g} \in \mathrm{SE}(3)$ 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 C, where each configuration $q \in C$ represents a particular set of joint angles for the manipulator. The collision-free subset of the configuration space is denoted as $Cfree \subseteq C$. The objective for the motion planner is to find a path from an initial configuration $q_{\text{start}} \in C_{\text{free}}$ to a goal configuration

$q_{\text{goal}} \in C_{\text{free}}$, ensuring that the chosen grasp T_g is reached while minimizing path length, avoiding obstacles, and adhering to the robot's kinematic constraints (Rudorfer et al., n.d.).

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 changes in the environment. Performance metrics such as planning time, path length 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 path length. 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. 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 different scenarios.
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**:

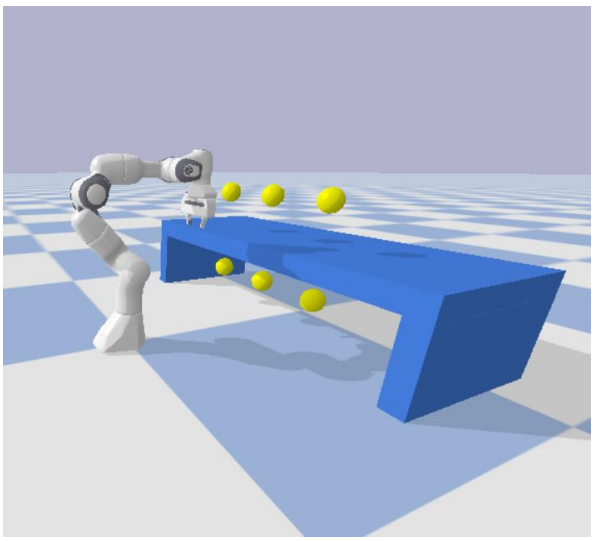

Figure 7 Pybullet environment with a robotic hand and 6 goal objects (Yellow) on the table and under the table

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 scenarios for performance evaluation in the first phase.

- **Jogramop Framework**:

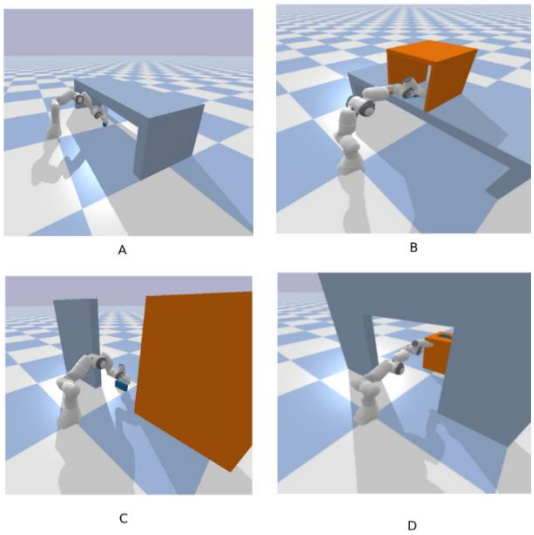

Figure 8 A robotic hand trying to grasp an object in 4 different scenes from Jogramop framework.

The Jogramop framework (Figure 5) was utilized in the second phase to provide a more structured and standardized set of benchmark scenarios. These 20 predefined scenarios in 4 different scenes simulate real-world challenges, including confined spaces and complex obstacle distributions. 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. 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. The uniformity of these scenarios ensured consistent testing and helped in comparing the algorithm’s performance across different conditions.

In summary, the combination of these algorithms and frameworks allowed for a comprehensive exploration of motion planning. 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 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.

4.5.2 Testing Scenarios

The algorithms were evaluated in two distinct environments. In the PyBullet environment, tests were conducted with goals placed at three positions—center, left, and right—both on top of and under a table, focusing on collision-free pathfinding in simpler setups. The Jogramop framework, on the other hand, presented more advanced benchmark scenarios with tight spaces and complex obstacle configurations, allowing for a deeper analysis of RRT*'s performance in more challenging environments.

4.5.3 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 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.

4.8 Overview of Results

The experimental setup was designed to assess the performance of four algorithms—JPlusRRT, 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, to get insights from 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.

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.

4.8.1 PyBullet Environment Results

Four distinct motion planning algorithms—JPlusRRT, RRTStar, IK-RRT, and BIK-RRT—were tested in the PyBullet environment across 100 trials each, evaluating their performance in two specific scenarios. In Scenario 1, the goal was placed on the table, while in Scenario 2, the goal was located under the table, introducing a more complex, constrained environment. The

results of these tests, including average planning times, path lengths, and success rates, are presented in Table 1.

*Table 1 PyBullet environment, 2 Scenarios, **(1)** Goal on the table, (**2**) Goal under the table, results for 4 algorithms, 100 trials.*

Algorithm	Avg Path Length (1)	Avg Planning Time (s) (1)	Avg Success Rate (1)	Avg Path Length (2)	Avg Planning Time (s) (2)	Avg Success Rate (2)
JPlusRRT	0.34533	0.00405	1.00	0.56543	0.00636	0.964
RRTStar	0.27263	0.00209	0.968	0.37234	0.00435	0.916
IKRRT	0.57062	0.46019	1.00	0.67172	0.55925	0.986
BIKRRT	2.64929	0.04571	1.00	2.84809	0.08928	0.924

JPlusRRT:

JPlusRRT demonstrated relatively fast planning times, particularly when the goals were placed on top of the table. The average path length on the table was 0.34533, indicating efficient exploration with reasonably short paths. However, while JPlusRRT maintained good speed, it was not the fastest algorithm overall. Planning times averaged 0.00405 seconds in the "On Table" scenario, and 0.00636 seconds when goals were placed under the table.

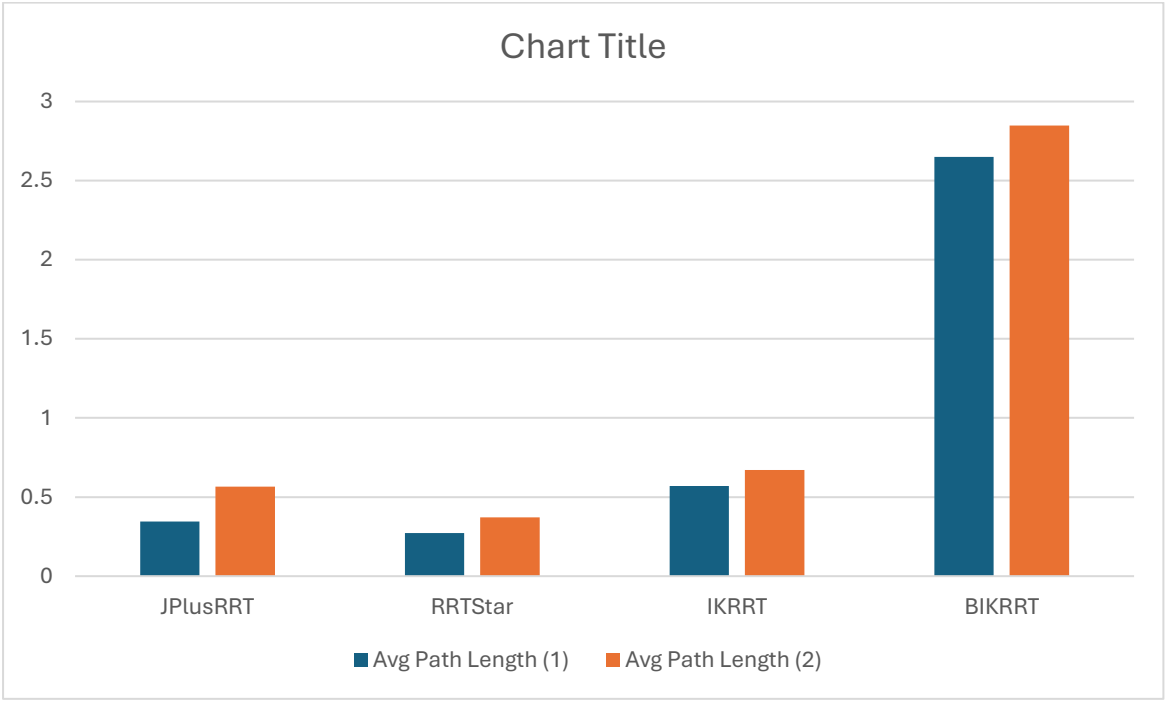

Figure 9 Average path length, (1) scenario 1 with the goal on the table, (2) scenario 2 with the goal under the table

Success rates remained consistently high at **100%** in the "On Table" scenario, with only a slight dip to **96.4%** when the goals were placed under the table. The more constrained environment affected its speed marginally, but JPlusRRT maintained strong adaptability and robustness in both scenarios.

RRTStar:

RRTStar demonstrated the fastest planning times in both scenarios, excelling in its ability to optimize paths while maintaining low computation times. On the table, the average planning time was 0.00209 seconds, and under the table, it was 0.00435 seconds. RRTStar also produced the shortest paths, with an average path length of 0.27263 on the table and 0.37234 under the table.

Although its success rate dipped slightly to **91.6%** under the table, RRTStar's performance overall remained strong, especially in its ability to handle more complex environments efficiently.

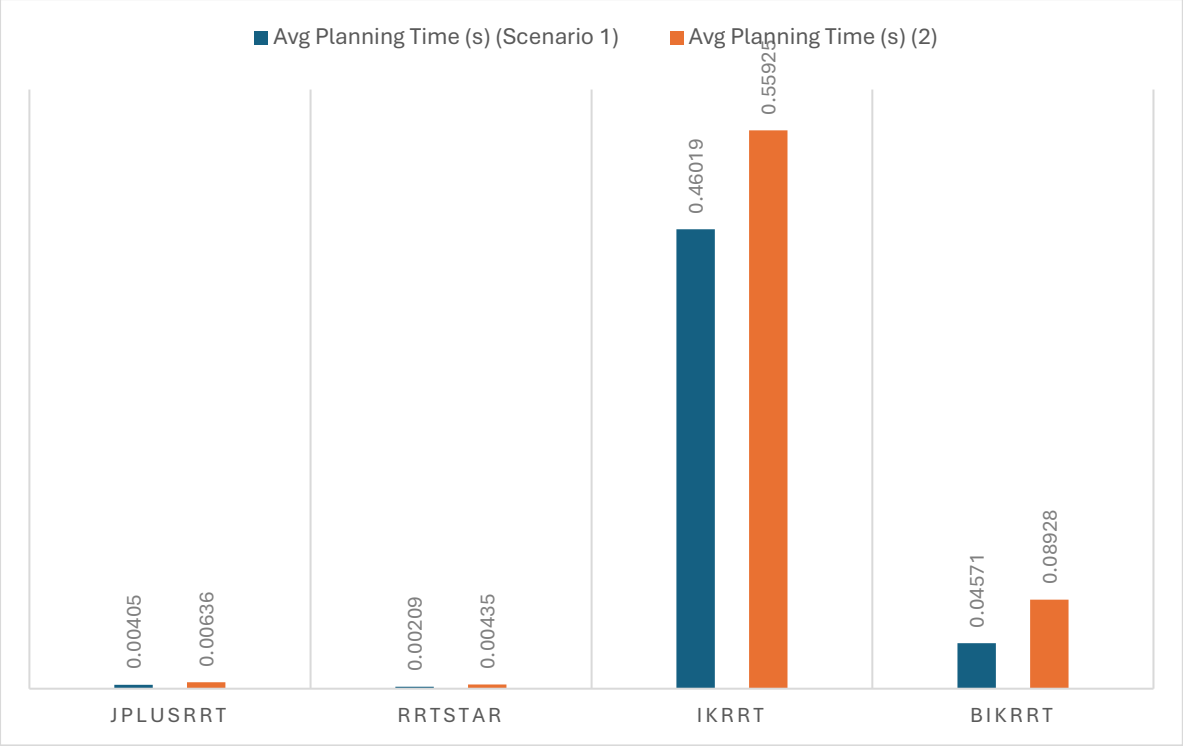

Figure 10 Average planning time, (1) scenario 1 with the goal on the table, (2) scenario 2 with the goal under the table

IK-RRT:

IK-RRT exhibited the longest planning times due to the complexity of solving inverse kinematic constraints. The average planning time was 0.46019 seconds on the table, which increased to 0.55925 seconds under the table.

Despite its slower performance, IK-RRT maintained a high success rate, with 100% on the table and 98.6% under the table. However, the longer path lengths—0.67172 under the table—suggest that while IK-RRT is reliable, it is significantly slower and produces longer paths in more complex environments.

BIK-RRT:

BIK-RRT showed the longest paths and the most inconsistent planning times across both scenarios. The average path length was 2.64929 on the table and 2.84809 under the table, indicating a tendency for over-exploration.

Planning times fluctuated significantly, especially under the table, with an average of 0.08928 seconds, reflecting variability in performance. While the success rate was 100% on the table, it dropped to 92.4% under the table, which highlights the challenges BIK-RRT faces in more constrained environments due to its bidirectional search method.

4.8.2 Jogramop Framework Results: 20 Benchmark Scenarios

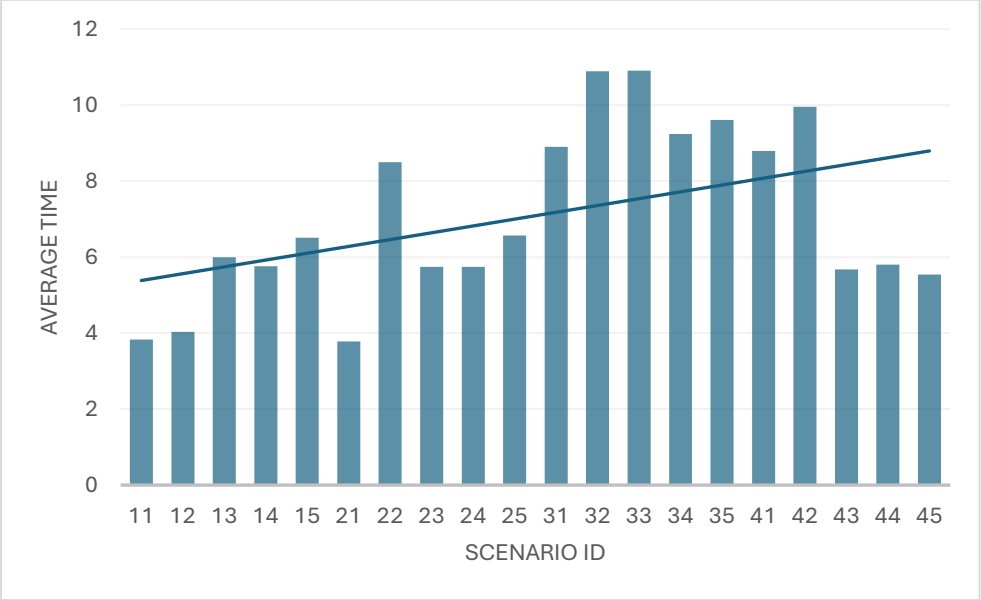

Figure 11 RRTStar algorithm average time results in Jogramop environment in 20 different scenarios, with trending line. Max iteration: 1000.

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.

4.8.3 Key Insights

RRTStar consistently demonstrated the fastest planning times across both scenarios, particularly in simpler environments like Scenario 1 (goal on the table). Its average planning time was 0.00209 seconds in Scenario 1 and 0.00435 seconds in Scenario 2 (goal under the table), making it the most efficient algorithm. Despite slightly lower success rates in tighter spaces (91.6% under the table), RRTStar consistently produced the shortest paths, with path lengths averaging 0.27263 in Scenario 1 and 0.37234 in Scenario 2.

JPlusRRT, while still fast, did not outperform RRTStar in terms of planning time. It averaged 0.00405 seconds in Scenario 1 and 0.00636 seconds in Scenario 2. JPlusRRT maintained robust success rates across both scenarios, slightly dipping to 96.4% under the table. Path lengths were slightly longer than RRTStar's, indicating JPlusRRT's focus on exploration speed rather than path optimization.

IK-RRT exhibited the longest planning times due to its need to solve inverse kinematic constraints. In Scenario 1, its average planning time was 0.46019 seconds, increasing to 0.55925 seconds in Scenario 2. However, it maintained a high success rate of 100% in Scenario 1 and 98.6% in Scenario 2, proving its reliability in finding valid paths, even though the paths were longer.

BIK-RRT was the most inconsistent, with planning times fluctuating significantly. It averaged 0.04571 seconds in Scenario 1 and 0.08928 seconds in Scenario 2. The algorithm's bidirectional search occasionally led to over-exploration, resulting in the longest paths in both scenarios—averaging 2.64929 on the table and 2.84809 under the table. Success rates remained at 100% in Scenario 1 but dropped to 92.4% in Scenario 2, further illustrating its challenges in more constrained environments.

These results underscore the diverse strengths and weaknesses of the tested algorithms. RRTStar is clearly the best choice for environments that prioritize fast, efficient planning, while JPlusRRT offers slightly slower times but a more adaptable approach. IK-RRT stands out for its high reliability in complex tasks, despite its slower performance, and BIK-RRT, while

capable, suffers from inconsistency and over-exploration, making it less suitable for environments where efficiency and precision are critical.

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.

During this project, grasp planning was tested both with and without dedicated planning to evaluate its impact on the overall performance of the algorithms. However, due to time constraints, a simplified grasping mechanism was ultimately implemented. In each scenario within the Jogramop framework, the system selected a grasp from a set of 200 predefined grasp candidates. Each grasp is a 4-dimensional array which the fourth array is the orientation. The algorithms (RRT*, IK-RRT, BIK-RRT) then planned the robot's inverse kinematics based on the chosen orientation. If a valid inverse kinematics solution was not found for the selected grasp, the algorithm automatically moved to the next available grasp position in the set, iterating until a feasible solution was identified. This approach allowed for efficient testing of grasping functionality within the constraints of the project.

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.. 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: Evaluation and Reflection

5.1 Achievement of the Project

This project successfully implemented and evaluated several motion planning algorithms—RRT, RRT*, JPlusRRT, IK-RRT, and BIK-RRT—within the context of integrated grasp and motion planning. The primary achievement lies in providing a comparative analysis of these algorithms across predefined scenarios, focusing on key performance metrics such as planning time, success rate, and path exploration efficiency. The study provides insights into how these algorithms perform under varying environmental configurations, contributing practical knowledge to the field of robotic motion planning.

5.2 Project Outcomes vs. Project Objectives

The project outcomes closely matched the stated objectives. By implementing the selected algorithms, the research achieved a clear understanding of their computational complexity, planning efficiency, and path-finding capabilities. Performance metrics like planning time, success rate, and path exploration efficiency were carefully analyzed across scenarios, providing detailed results.

While the project did not fully integrate real-world grasping tasks, it successfully focused on evaluating motion planning performance, which was the central goal. The insights gained lay the groundwork for further research that integrates grasp and motion planning in more complex scenarios.

5.3 Algorithm Selection

RRT*, JPlusRRT, IK-RRT, and BIK-RRT were chosen specifically for their distinct contributions to motion planning. RRT* was selected for its ability to find optimal paths by continuously refining the tree structure. Its asymptotic optimality makes it suitable for tasks where efficiency and optimality in path planning are crucial. RRT* was a clear choice because of its well-

established balance between exploration and optimization, making it effective for both basic and more constrained environments.

JPlusRRT was chosen for its focus on speed and adaptability, making it a good counterpoint to RRT*. Unlike RRT*, JPlusRRT prioritizes rapid exploration over path optimality, which can be advantageous in scenarios where speed is more critical than precision.

IK-RRT and BIK-RRT were selected due to their focus on handling inverse kinematics. While these algorithms are computationally heavier, they offer precise motion planning that is essential when dealing with robotic arms in constrained spaces. The algorithms were chosen specifically to explore scenarios where robotic configuration constraints are key to finding valid motion plans. Other algorithms, such as PRM, were not selected because they are designed for different motion planning paradigms (e.g., roadmap-based approaches) and do not directly align with the specific needs of grasp and motion planning in this context.

5.4 Evidence: Usability Test, Performance Studies

The performance of each algorithm was evaluated through comprehensive tests conducted across both the PyBullet and Jogramop environments, revealing distinct differences in their ability to handle the varying challenges posed by the scenarios:

- **RRTStar consistently provided the shortest, optimized paths** in both environments, particularly excelling in Scenario 1 (goal on the table). However, this came at the cost of longer planning times, especially in more complex, obstacle-rich environments like Scenario 2 (goal under the table).
- **JPlusRRT excelled in terms of planning time**, delivering rapid, feasible paths with average planning times significantly faster than the other algorithms. However, the quality of these paths, while sufficient, was not always optimal, as seen in slightly longer path lengths compared to RRTStar.
- **IK-RRT showed longer planning times** due to the complexity of solving inverse kinematics, but it maintained the highest success rates across both scenarios, proving its reliability for tasks requiring precision in path planning despite the slower performance.
- **BIK-RRT demonstrated the most variability**, with inconsistent planning times and longer path lengths, particularly in Scenario 2. While its bidirectional search allowed it to

explore more thoroughly, this often led to over-exploration, reducing its overall efficiency and success rate in constrained environments.

These results underscore the balance between speed, path quality, and success rate across the algorithms, with RRTStar favoring path optimization, JPlusRRT excelling in speed, and IK-RRT proving reliable for precision tasks. BIK-RRT, though capable, struggled with consistency in more complex settings.

5.5 Comparison Between the Chosen Option and Alternatives

The choice of these specific algorithms provided a comprehensive evaluation of motion planning performance, with each algorithm offering distinct advantages. RRTStar was particularly suited for scenarios where path optimality was the priority, delivering the most efficient paths even in complex environments. JPlusRRT, on the other hand, proved more effective for scenarios requiring fast and efficient planning, excelling in environments where quick exploration and adaptation were essential. IK-RRT and BIK-RRT demonstrated their value in situations demanding kinematic precision, with IK-RRT especially standing out for its high success rates despite longer planning times.

The research highlighted that while each algorithm had unique strengths, there was no universal solution that could address all motion planning challenges. The selection of the most appropriate algorithm depended heavily on the specific task requirements and environmental conditions. For example, RRTStar's focus on path optimization made it the best choice for tasks requiring highly efficient navigation, while JPlusRRT's speed made it more suited for real-time applications where rapid decision-making was critical.

Other algorithms, such as PRM or A*, were not selected because their approaches did not align as well with the needs of real-time, adaptive motion planning. In scenarios requiring continuous refinement and optimization (as offered by RRTStar) or fast exploration (as in JPlusRRT), these alternative methods lacked the adaptability and dynamic response required for handling the complexity of the environments studied.

5.6 Areas for Improvement and Future Steps

Several areas for improvement emerged during the research, highlighting potential directions for enhancing the evaluation of the algorithms. First, the scenario complexity can be expanded

to include more intricate environmental layouts or multi-task planning to better assess the full capabilities of the algorithms. Additionally, algorithm optimization is a critical area, as parameters such as step size and sampling strategies could be fine-tuned to reduce planning time and improve path quality. Lastly, integrating grasping tasks into the analysis would provide a more complete picture of the algorithms' suitability for real-world robotic applications, as the current study focused solely on motion planning.

To address these areas, future research should focus on the following:

1. Refining Scenario Complexity**:** Introducing more diverse, real-world-like scenarios that challenge the algorithms to navigate more intricate environments will allow for a more comprehensive evaluation.
2. Optimizing Algorithm Parameters**:** Fine-tuning parameters like goal bias, step size, and neighbour search radius can enhance the efficiency and performance of algorithms such as RRT* and JPlusRRT.
3. Extending to Grasp Planning**:** Incorporating more advanced grasp planning into future experiments will provide a broader understanding of the algorithms' capabilities, especially in complex robotic manipulation tasks.

By addressing these areas, future research can further refine the efficiency, adaptability, and applicability of motion planning algorithms in both simulated and real-world environments.

Chapter 6: 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.

6.1 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.

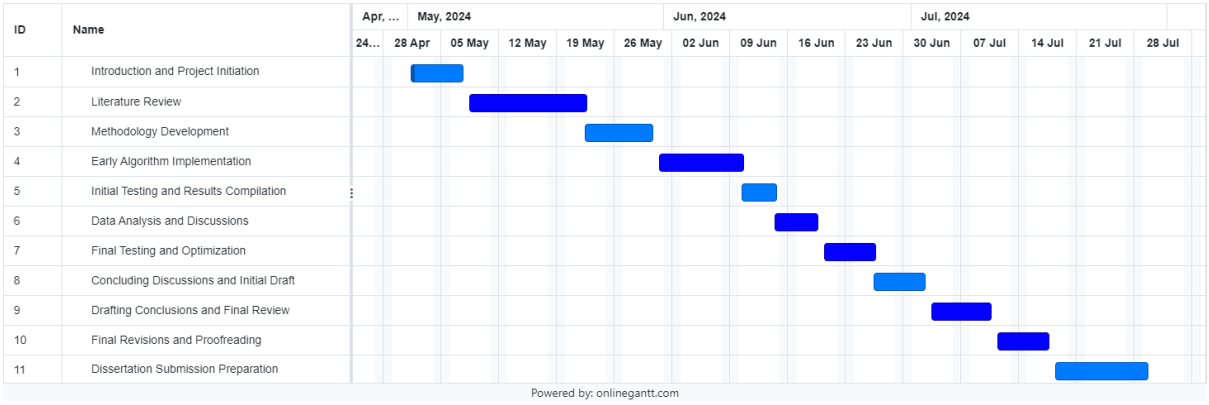

Figure 12 Gantt Chart for Dissertation on Integrated Grasp and Motion Planning.

Initially, according to the timeline presented in Figure 12, the project was expected to progress at a faster pace, with key milestones such as the early algorithm implementation and data analysis expected to be completed earlier. However, as the project advanced, it became evident that some tasks, particularly the testing and optimization phases, required more time than originally planned. The additional complexity of integrating and refining the algorithms, as well as analyzing the results, extended the project timeline beyond the initial expectations.

Consequently, tasks such as final testing and drafting the conclusions took longer to complete, pushing the final dissertation submission preparations to a later date.

6.2 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.

6.3 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.

6.4 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.

6.5 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.

6.6 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.

6.7 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 7: Business Strategy and Market Integration

7.1 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.

7.2 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.

7.3 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 2 SWOT/TOWS (Weihrich, 1982; Wheelen and Hunger, 2008) 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	**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 • Collaboration potential with academic institutions and industry leaders • Untapped market segments in healthcare, manufacturing, and logistics	• Leverage superior technology to enter high-growth industries like healthcare and logistics. • Form partnerships with leading robotics manufacturers to scale production and expand market reach.	• Collaborate with research institutions to reduce development costs and gain access to technical expertise. • Build brand recognition through strategic marketing in niche sectors.
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. • 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. • Implement a phased rollout to mitigate risks associated with market fluctuations or rapid technological changes.

7.4 Competitive Advantage

Figure 7 highlights the system's competitive advantage through Porter's Five Forces. The integrated grasp and motion planning system stands out in the robotics marketplace by combining both processes into a unified framework. This seamless integration offers superior performance in complex, obstacle-rich environments, making it particularly valuable in sectors like healthcare and manufacturing, where precision is key.

Key advantages include:

- **Superior Performance**: The system handles intricate environments effectively, offering precision in tasks like surgery or delicate assembly.
- **Real-time Adaptability**: The system dynamically adjusts to changes, crucial in industries like logistics where conditions fluctuate rapidly.
- **Precision and Efficiency**: By minimizing errors in both grasping and motion, the system reduces operational risks and boosts productivity.

These features give the system a clear advantage over competitors, positioning it as a leader in industries where efficiency and adaptability are crucial. Unlike competitors, which often focus on either motion planning or grasping, our solution optimizes both simultaneously in real time, setting it apart in addressing complex tasks efficiently.

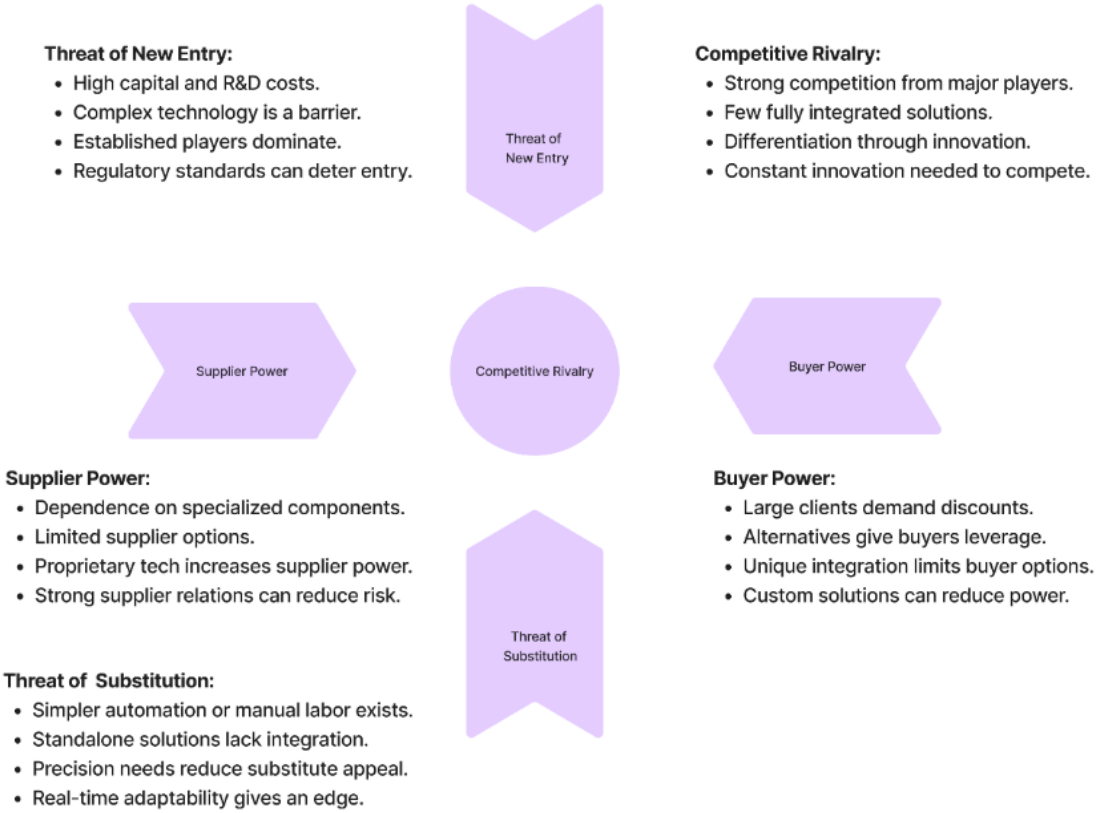

Figure 13 Porter's Five forces analysis of Integrated grasp and motion planning.

7.5 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.

7.6 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 8: Conclusion

Summary of Research

The primary objective of this research was to implement and evaluate several motion planning algorithms—specifically RRT, RRT*, JPlusRRT, IK-RRT, and BIK-RRT—in the context of integrated grasp and motion planning tasks. The study focused on assessing these algorithms based on key performance metrics such as planning time, success rate, and path exploration efficiency. By testing these algorithms across various predefined scenarios in both the PyBullet and Jogramop environments, the research explored how each algorithm manages different motion planning challenges, particularly in terms of computational efficiency and the quality of the generated paths. Additionally, this comparison provided insights into how adaptable each algorithm is to diverse environmental configurations.

Key Findings

The results demonstrated that RRT* excelled in generating optimal paths, particularly in environments with moderate complexity, such as when goals were on top of the table. However, in more complex or confined spaces, like goals placed under the table, the computational cost increased significantly due to the rewiring process aimed at optimizing the paths. Despite the increased computational load, RRT* maintained a robust success rate and produced feasible paths across most scenarios, although its planning time was longer than faster algorithms like JPlusRRT. JPlusRRT showed faster adaptability, especially in environments with fewer obstacles, while IK-RRT maintained a high success rate but at the cost of longer planning times.

Contributions to the Field

By comparing the performance of RRT*, JPlusRRT, IK-RRT, and BIK-RRT in both simple and complex environments, this study contributes to the ongoing effort to balance computational efficiency with optimal path generation. The integration of RRT* into the Jogramop framework offers a deeper understanding of how the algorithm performs in more confined and complex spaces, enriching the body of research on motion planning algorithms for robotic applications.

Practical Applications

The findings from this study have practical implications for industries such as manufacturing, healthcare, and autonomous systems. RRT*, with its emphasis on path optimization, is highly applicable to tasks requiring precise and efficient navigation, such as in robotic arms used in manufacturing or in assistive devices in healthcare. The insights gained from this research, particularly around how algorithms like JPlusRRT balance speed and path quality, can be applied to real-time autonomous systems, such as self-driving vehicles or robots that operate in dynamic environments.

Ethical Considerations

During this research, no human subjects or personal data were involved. However, several ethical aspects were considered to ensure that the development and implementation of motion planning algorithms align with responsible practices. Although no personal data or human subjects were involved in the study, the potential real-world applications of these algorithms necessitate attention to key ethical concerns.

First, data privacy and security is paramount in robotics, especially in systems that operate in environments like healthcare or autonomous vehicles, where sensor data is collected. Although this study focused on simulations, ensuring that such data is handled securely in real-world applications is essential to comply with privacy regulations and protect sensitive information.

Safety is another critical ethical consideration, particularly when implementing robots in human-centered environments. While this research used simulated environments, deploying robots in physical spaces introduces the risk of accidents or harm. Algorithms like those explored in this study must prioritize safety by ensuring collision-free paths and reliable navigation, minimizing risks to humans and property in real-world deployments.

Additionally, bias in algorithm design must be acknowledged. Although the research did not deal with biased data, future implementations of these algorithms should account for the risk of biases emerging in decision-making, particularly in dynamic or unstructured environments. Continuous evaluation and adaptation are necessary to prevent disproportionate outcomes or unfair treatment of certain users or conditions.

The environmental impact of robotic systems also warrants ethical consideration. While robots can enhance efficiency and sustainability in industries such as manufacturing, their production and energy consumption must be managed responsibly. The motion planning algorithms

studied here should be optimized for energy efficiency, contributing to the broader goal of minimizing the environmental footprint of robotics.

Finally, the socioeconomic impact of robotics, particularly the potential for job displacement, raises ethical concerns. Automation in industries such as logistics and manufacturing may reduce the demand for certain types of labor. Ethical development and deployment of robots should consider the importance of reskilling and retraining workers whose roles may be affected, ensuring that technological progress benefits society.

Final Thoughts

This research provided a comprehensive exploration of the strengths and weaknesses of RRT*, JPlusRRT, IK-RRT, and BIK-RRT in different environments. Despite challenges related to the complexity of setting up environments and time limitations for testing, the study delivered meaningful results. Future work could focus on further optimizing these algorithms, testing hybrid approaches, adding path pruning techniques, and expanding evaluations to more dynamic real-world scenarios. By continuing this line of research, future studies can help refine motion planning strategies and improve the practical application of these algorithms in real-world robotic systems.

References

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

Bicchi, A., 1995. On the Closure Properties of Robotic Grasping. Int J Rob Res 14. https://doi.org/10.1177/027836499501400402

Bohg, J., Morales, A., Asfour, T., Kragic, D., 2014. Data-driven grasp synthesis-A survey. IEEE Transactions on Robotics 30. https://doi.org/10.1109/TRO.2013.2289018

Calandra, R., Owens, A., Jayaraman, D., Lin, J., Yuan, W., Malik, J., Adelson, E.H., Levine, S., 2018. More than a feeling: Learning to grasp and regrasp using vision and touch. IEEE Robot Autom Lett 3. https://doi.org/10.1109/LRA.2018.2852779

Ciocarlie, M.T., Allen, P.K., 2009. Hand posture subspaces for dexterous robotic grasping. International Journal of Robotics Research 28. https://doi.org/10.1177/0278364909105606

Elbanhawi, M., Simic, M., 2014. Sampling-based robot motion planning: A review. IEEE Access. https://doi.org/10.1109/ACCESS.2014.2302442

Falcone, P., Borrelli, F., Asgari, J., Tseng, H.E., Hrovat, D., 2007. Predictive active steering control for autonomous vehicle systems. IEEE Transactions on Control Systems Technology 15. https://doi.org/10.1109/TCST.2007.894653

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

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

Fontanals, J., Dang-Vu, B.A., Porges, O., Rosell, J., Roa, M.A., 2015. Integrated grasp and motion planning using independent contact regions, in: IEEE-RAS International Conference on Humanoid Robots. https://doi.org/10.1109/HUMANOIDS.2014.7041469

Garrett, C.R., Chitnis, R., Holladay, R., Kim, B., Silver, T., Kaelbling, L.P., Lozano-Perez, T., 2021. Integrated Task and Motion Planning. Annu Rev Control Robot Auton Syst. https://doi.org/10.1146/annurev-control-091420-084139

Ichnowski, J., Avigal, Y., Satish, V., Goldberg, K., 2020. Deep learning can accelerate grasp-optimized motion planning. Sci Robot 5. https://doi.org/10.1126/scirobotics.abd7710

Kahn, G., Villaflor, A., Ding, B., Abbeel, P., Levine, S., 2018. Self-Supervised Deep Reinforcement Learning with Generalized Computation Graphs for Robot Navigation, in: Proceedings - IEEE International Conference on Robotics and Automation. https://doi.org/10.1109/ICRA.2018.8460655

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

Karaman, S., Walter, M.R., Perez, A., Frazzoli, E., Teller, S., 2011. Anytime motion planning using the RRT, in: Proceedings - IEEE International Conference on Robotics and Automation. https://doi.org/10.1109/ICRA.2011.5980479

Kavraki, L.E., Švestka, P., Latombe, J.C., Overmars, M.H., 1996. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. IEEE Transactions on Robotics and Automation 12. https://doi.org/10.1109/70.508439

Kopicki, M., Detry, R., Adjigble, M., Stolkin, R., Leonardis, A., Wyatt, J.L., 2016. One-shot learning and generation of dexterous grasps for novel objects. International Journal of Robotics Research 35. https://doi.org/10.1177/0278364915594244

Kopicki, M., Zurek, S., Stolkin, R., Mörwald, T., Wyatt, J., 2011. Learning to predict how rigid objects behave under simple manipulation, in: Proceedings - IEEE International Conference on Robotics and Automation. https://doi.org/10.1109/ICRA.2011.5980295

Lavalle, S.M., 2006. PLANNING ALGORITHMS.

LaValle, S.M., 1998. Rapidly-Exploring Random Trees: A New Tool for Path Planning. In 129.

Leu, J., Cheng, Y., Liu, C., Tomizuka, M., 2022. Robust Task Planning for Assembly Lines with Human-Robot Collaboration.

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

Mahler, J., Liang, J., Niyaz, S., Laskey, M., Doan, R., Liu, X., Ojea, J.A., Goldberg, K., 2017. Dex-Net 2.0: Deep learning to plan Robust grasps with synthetic point clouds and analytic grasp metrics, in: Robotics: Science and Systems. https://doi.org/10.15607/rss.2017.xiii.058

Mishra, B., Schwartz, J.T., Sharir, M., 1987. On the existence and synthesis of multifinger positive grips. Algorithmica 2. https://doi.org/10.1007/BF01840373

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

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

Qureshi, A.H., Simeonov, A., Bency, M.J., Yip, M.C., 2019. Motion planning networks, in: Proceedings - IEEE International Conference on Robotics and Automation. https://doi.org/10.1109/ICRA.2019.8793889

Rudorfer, M., Hartvich, J., Vonásek, V., n.d. A Framework for Joint Grasp and Motion Planning in Confined Spaces.

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

Vahrenkamp, N., Berenson, D., Asfour, T., Kuffner, J., Dillmann, R., 2009. Humanoid motion planning for dual-arm manipulation and re-grasping tasks, in: 2009 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2009. https://doi.org/10.1109/IROS.2009.5354625

Vahrenkamp, N., Do, M., Asfour, T., Dillmann, R., 2010. Integrated grasp and motion planning, in: Proceedings - IEEE International Conference on Robotics and Automation. https://doi.org/10.1109/ROBOT.2010.5509377

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

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

Weihrich, H., 1982. The TOWS matrix-A tool for situational analysis. Long Range Plann 15. https://doi.org/10.1016/0024-6301(82)90120-0

Wheelen, T.L., Hunger, J.D., 2008. Strategic management and business policy - Acheiving Sus, Policy.

Wu, Zhenping et al. (2021) Fast-RRT: A RRT-based optimal path finding method, Applied Sciences (Switzerland), 11(24). Available at: https://doi.org/10.3390/app112411777.

Yan, W., Deng, Z., Chen, J., Nie, H., Zhang, J., 2019. Precision Grasp Planning for Multi-Finger Hand to Grasp Unknown Objects. Robotica 37. https://doi.org/10.1017/S0263574719000031

Yi, J., Yuan, Q., Sun, R., Bai, H., 2022. Path planning of a manipulator based on an improved P_RRT* algorithm. Complex and Intelligent Systems 8. https://doi.org/10.1007/s40747-021-00628-y

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

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

Ziegler, J., Bender, P., Schreiber, M., Lategahn, H., Strauss, T., Stiller, C., Dang, T., Franke, U., Appenrodt, N., Keller, C.G., Kaus, E., Herrtwich, R.G., Rabe, C., Pfeiffer, D., Lindner, F., Stein, F., Erbs, F., Enzweiler, M., Knoppel, C., Hipp, J., Haueis, M., Trepte, M., Brenk, C., Tamke, A., Ghanaat, M., Braun, M., Joos, A., Fritz, H., Mock, H., Hein, M., Zeeb, E., 2014. Making bertha drive-an autonomous journey on a historic route. IEEE Intelligent Transportation Systems Magazine 6. https://doi.org/10.1109/MITS.2014.2306552

Appendix

All the project codes related to this research are available at the following public repository: https://github.com/jhiyar/aston-ai-igmp.