Dynamically-feasible real-time local planning for fast outdoor robots

by Nguyen Thanh Trung Le

Thesis submitted in fulfilment of the requirements for the degree of

Master of Research

under the supervision of Dr. Graeme Best

University of Technology, Sydney
Faculty of Engineering and Information Technology

July 2024

Certificate of Original Authorship

I, Nguyen Thanh Trung Le declare that this thesis, is submitted in fulfilment of the requirements for the award of Master of Research in Computer Science, in the School of Mechanical and Mechatronic Engineering at the University of Technology Sydney.

This thesis is wholly my own work unless otherwise referenced or acknowledged. In addition, I certify that all information sources and literature used are indicated in the thesis.

This document has not been submitted for qualifications at any other academic institution.

This research is supported by the Australian Government Research Training Program.

Signature: Production Note:
Signature removed prior to publication.

Date: 17/07/2024

Dynamically-feasible real-time local planning for fast outdoor robots

by

Nguyen Thanh Trung Le

A thesis submitted in fulfilment of the requirements for the degree of Master of Research

Abstract

In recent years, applications from autonomous navigation of ground mobile robots have been pivotal in exploring the advantages of deploying autonomous robots optimised for outdoor navigation. This approach holds potential across various industries, with a key strength in their ability to navigate efficiently through unstructured terrains. For robots operating in outdoor environments, autonomous navigation plays a major role in defining the effectiveness of an operation. The objective is to enable the platform to autonomously and securely navigate towards the desired destination safely and efficiently. However, this feature comes with challenges and limitations for operators. Factors including obstacle avoidance, the efficiency and reliability of the path planning solution, and the consideration of the platform's dynamics are the major problems. This thesis introduces a novel local path-planning technique for high-speed off-road robotic vehicles called Adaptive Trajectory Library (ATL). The method integrates a dynamic trajectory library, filtered from the platform's physical characteristics to match with operation-specified waypoints. The planner ensures the dynamic feasibility of the target platform by implementing pre-defined velocity configurations extracted from the robot's performance data. ATL can effectively search for feasible trajectories to adapt to the robot's current state, enhancing its motion's smoothness. Experimental results demonstrate the controller's response and effectiveness in maintaining dynamic feasibility at speeds up to 5m/s, showcasing its potential for improving the performance of fast off-road robotic vehicles in practical environments.

Acknowledgements

I extend my appreciation to the entire academic community at the University of Technology Sydney (UTS) for providing a conducive learning environment. I would like to thank my supervisor, Dr Graeme Best and co-supervisor, Prof. Robert Fitch, whose unwavering support has been instrumental in the completion of this thesis. My sincere thanks go to my colleagues and fellow researchers, Dr. Edward Bray and Dr. Ki Myung Brian Lee, who have shared their insights and experiences throughout this academic journey. The collaborative spirit within the research community has greatly enriched my understanding and has been a source of inspiration.

I am grateful to my friends and family for their unwavering support and understanding during the challenging phases of this thesis. Their encouragement and belief in my abilities have been a driving force, and I am truly fortunate to have such a strong support system.

Special thanks to the staff at the Mechanical Mechatronic and Robotic (MMR) department, whose guidance and resources have been pivotal to my academic growth and research endeavors.

Nguyen Thanh Trung Le Sydney, Australia, 2024.

Contents

Chapter 1

Introduction

Deploying autonomous robots optimised for outdoor navigation presents tremendous potential across various industries. Their advantage lies in their capacity to travel through complex terrain areas efficiently. Yet, while the prospects are promising, integrating these high-speed robots brings unique challenges that set them apart from conventional indoor, low-speed systems. The transition to outdoor operation necessitates a thorough reassessment of navigation approaches, obstacle detection, avoidance methods, and the overall robustness of the system. This chapter motivates the need for a path-planning solution for autonomous robots operating at high speed, under the conditions of unstructured outdoor environments. This chapter also includes and presents the research objectives and principal contribution.

1.1 Challenges of Path Planning for Off-road Robotic Platform

Designing and planning for a fast off-road robotic platform introduces challenges that require careful consideration and innovative solutions. Considering an operation that requires a path planning solution for an off-road robotic platform, we identify three main challenges: unstructured outdoor environments, efficiency in computational usages, and kinodynamic planning.

FIGURE 1.1: Outdoor Robotic Platform designed for operation in challenging outdoor environments. The FORV (Fast Off-road Vehicle) developed at UTS, is the experimental platform used in this work.

Deploying an autonomous platform in **unstructured outdoor environments** presents challenges and complexities. The nature of outdoor terrains introduces different unexpected scenarios, ranging from collisions with unidentified obstacles to the acquisition of noisy data, which can directly impact the overall quality of autonomous navigation. The presence of unknown obstacles (Melchior and Simmons, 2007), remains a constant challenge for mobile robots. A robot may encounter obstacles not included in its initial path planning in environments. These could be moving objects, temporary obstructions, or obstacles obscured by other environmental factors. On the other hand, high-speed moving objects (Frazzoli et al., 2001) generate a unique set of challenges for mobile robots. The unpredictable nature of these dynamic obstacles necessitates instantaneous decision-making on the platform, ensuring the safety of both the robot and the surrounding environment.

Efficiency in computational usage and real-time performance directly contribute to the quality of an autonomous navigation application with the outdoor ground robots, especially those who are capable of travelling at high speed. Outdoor robots, by design, cover vast outdoor areas and provide the platform with the capability to travel at high speeds. While this is advantageous regarding efficiency, it simultaneously challenges the autonomous navigation system. This challenge revolves around the system's ability to

rapidly and reliably control the platform's movements, ensuring it navigates smoothly while avoiding static and dynamic obstacles. The crux of the matter lies in the demand for a computational and efficient solution that can operate in real-time, effectively interpreting and responding to the conditions of the surrounding environment.

Nevertheless, the field of path planning for mobile robots includes **kinodynamic planning**, a subfield of motion planning in robotics that specifically deals with the planning of robot motions while considering both the kinematic and dynamic constraints of the system. Motion planning involves determining a feasible path for a robot to reach its goal while avoiding environmental obstacles. Kinodynamic planning goes beyond kinematic planning by considering the robot's dynamics, including acceleration, velocity, and forces during real-time operation (Donald et al., 1993).

Furthermore, extreme climatic conditions (Chen et al., 2012) can significantly impact a mobile robot's performance. Extreme weather conditions can lead to mechanical stress, sensor malfunctions, or temporary immobilisation. Additionally, complex terrains (Singh et al., 2018) present another notable problem.

In conclusion, the design and planning of fast off-road robotic platforms demand a comprehensive approach to overcoming the challenges of unstructured outdoor terrain, efficiency in computational usage and kinodynamic planning. As technology evolves, integrating advanced sensors, adaptive mobility systems, and real-time decision-making capabilities are crucial in enhancing the resilience and effectiveness of fast off-road robotic platforms in dynamic and unpredictable environments.

1.2 Path Planning for Mobile Robots

Robots operating autonomously in real-world environments, both indoors and outdoors, must be able to navigate their environment safely. Robot path planning is a critical aspect of robotics that involves determining the optimal route for a robot to navigate from its current position to a specified destination while avoiding obstacles.

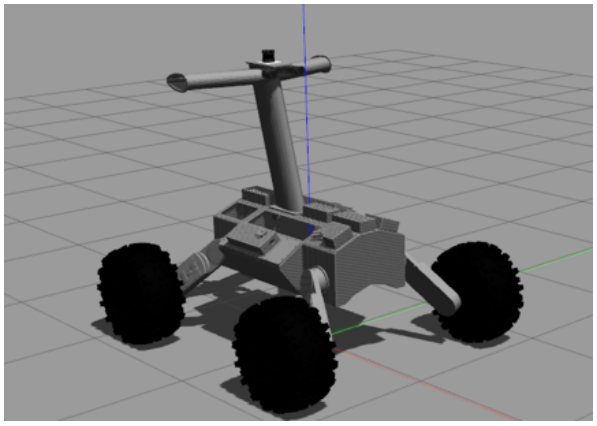


FIGURE 1.2: The Fast Off-Road Vehicle (FORV) developed at UTS in (a) simulation and (b) hardware, used in simulation and hardware experiments.

Path planning algorithms aim to address the challenges generated by unstructured environments, platform dynamics and the need to optimise parameters such as time, computational efficiency, or safety. These algorithms can be categorised into various types, including grid-based methods, probabilistic roadmaps, potential fields, and machine learning-based approaches. The ultimate goal is to equip robots with the ability to autonomously and adaptively navigate through complex and unpredictable surroundings, enabling them to perform tasks efficiently and securely in real-world scenarios.

Modern robotic systems generally employ a decision-making hierarchy in response to these challenges. This hierarchical approach comprises four distinct phases, outlined as follows:

- **High-Level Decision Making**: The highest level of the hierarchy, this phase defines the tasks for the platform, outlining the overall objective to be achieved.
- **Global Planner**: This phase generates waypoints to guide the platform, aligning with the specified tasks and objectives produced during high-level decision-making.
- **Local Planner**: Tasked with processing waypoints received from the global planner, the local planner computes the necessary velocities for the robot. It ensures collision-free navigation and considers the dynamic characteristics of the platform to calculate movement toward the commanded destinations.
- **Controller**: Operating at the lowest level, this phase manages the platform's movement based on the velocities commanded by the local planner.

This thesis focuses on developing a Local Planner solution for outdoor ground robots. Local path planning is crucial for mobile robotics as it enables real-time adaptability to the environments, allowing robots to navigate around obstacles and unforeseen challenges. Without effective local path planning, robots may struggle to avoid collisions and efficiently reach their destinations, limiting their overall autonomy and operational effectiveness. Local path planning for a high-speed outdoor robotic vehicle such as the one in Figure 1.2 operating in an unstructured outdoor environment is more challenging than traditional indoor robots (Likhachev and Ferguson, 2009), as outdoor operation presents difficulties not experienced in structured indoor arenas, such as unpredictable terrain and changes to the environment during operation (Chu et al., 2012; Goswami, 2017).

Gathering real-time information from the platform's sensors, local planning techniques adjust the movements to navigate the target platform through dynamic obstacles and ensure the platform's performance among many challenges of a path planning operation. This plays a crucial component of autonomous navigation, where a robot must decide how to reach a given goal location while adapting certain constraints, such as avoiding obstacles or choosing a short route (Karur et al., 2021). The local path planner chooses which trajectory the robot should follow at any instant (Coulter, 1992b; Fox et al., 1997; Rösmann et al., 2012).

As robotics advances, the ongoing improvement of path planning strategies will play a major role in unlocking the full potential of autonomous robots in diverse and challenging real-world scenarios. The pursuit of faster, more responsive, and adaptable path-planning algorithms remains integral to the evolution of robotics and its applications in outdoor environments.

1.3 Research Objectives

This thesis is aimed to provide answers to following questions:

How can we design an efficient algorithm to enable a fast mobile robot to navigate autonomously?

Designing a path-planning solution for a complex environment such as unstructured outdoor terrain demands a trade-off compared to a general path-planning solution for indoor robots. Given the terrain's conditions, robots can travel longer distances and faster than other types of operation. Hence, the three crucial factors, outdoor unstructured terrain conditions, the platform's dynamic, and the efficiency in using computational resources, present the major challenging issues for local outdoor robot planning. By prioritising these aspects during the development of the new technique, the new planner can efficiently navigate the robotic platform through obstacles while ensuring the platform's dynamics characteristics.

What strategies can be employed to ensure the dynamic constraints of the platform are satisfied during the operating stage?
Satisfying the dynamic constraints of the robot has always been a problem for traditional path planning, where most techniques focus on identifying the shortest and obstacle-free route. Traditional algorithms (DWA (Christian Connette, 2023a), TEB (Rösmann, 2023)) can generate the needed control commands for the target platform to follow certain paths, yet, these values can't effectively handle the physical constraint of the platform during its movements across the environment. A library that can store a list of feasible velocities and accelerations of the platform and turn them into offline trajectories that match the demanded path can significantly increase the robot's performance.

1.4 Principal Contributions

This thesis makes a significant contribution by introducing an innovative approach to local planning for mobile robots, offering a comprehensive and efficient solution for path planning. This thesis proposes a new local planner for a fast outdoor robotic vehicle, called *Adaptive Trajectory Library* ATL, as shown in Figure 1.3. The ATL is demonstrated as the robot navigates towards the commanded red-circled waypoint. Lidar detects obstacles, generating a 2D costmap (depicted in black). Within this map, a library of dynamically feasible trajectories is analysed, discarding trajectories leading to collisions (depicted in red). Evaluating a cost function for collision-free trajectories (depicted in green), the robot follows the trajectory with the lowest cost (highlighted in thick green).

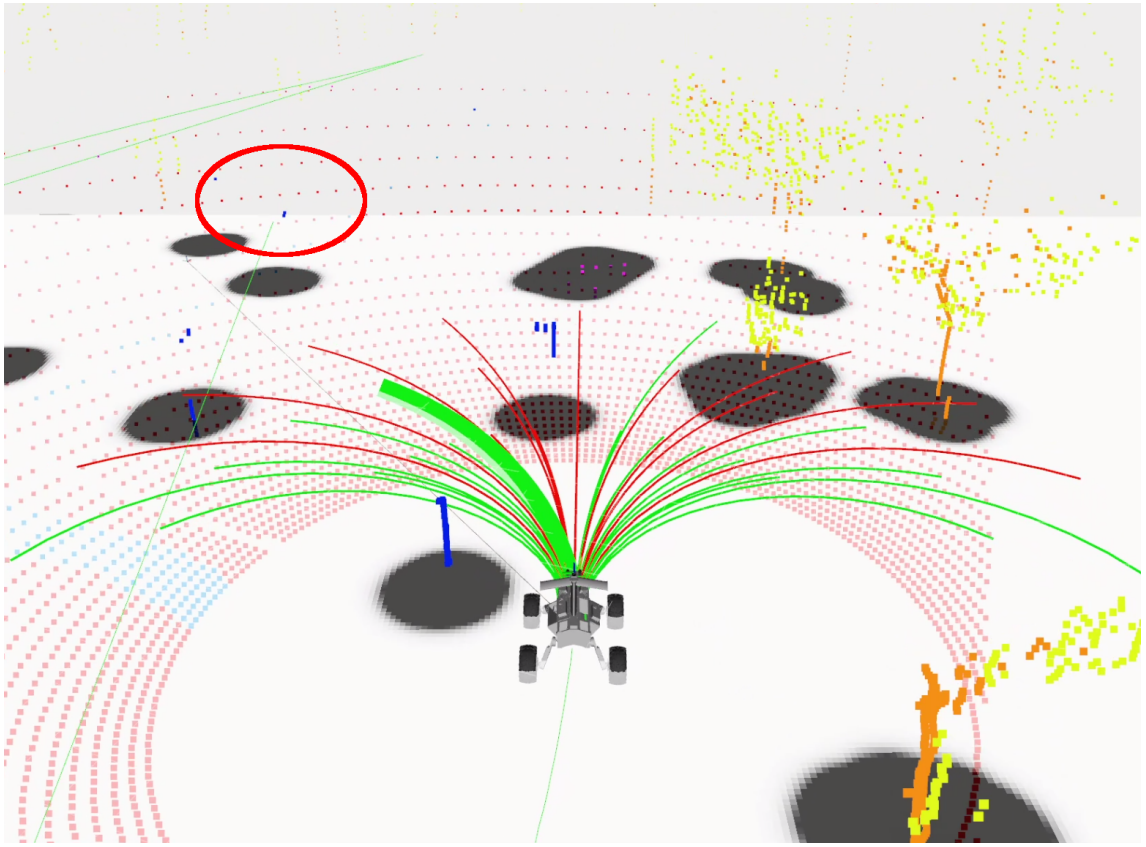


FIGURE 1.3: *Adaptive Trajectory Library* (ATL) in action as the robot moves to the red circled waypoint. Obstacles are detected by a lidar and represented on a 2D costmap (black). A dynamically feasible subset of the library is considered, and trajectories that result in collision are discarded (red). A cost function is evaluated for the collision-free trajectories (green), and the robot moves along the one with the lowest cost (thick green).

A library of feasible trajectories is first precomputed, containing combinations of linear and angular velocities that the robot can achieve. At each planning instant, a subset of the library is considered which is adapted to contain only those trajectories that are feasible to achieve given the current state of the robot. A trajectory is selected from this subset that is the most suitable at this time, considering the goal location and obstacle avoidance. This approach manages computational resources more efficiently than calculating possible trajectories online while ensuring that selected actions are feasible. Therefore, the robot's responsiveness is increased compared to other methods, even when travelling at high speed.

Implementing an offline trajectories library for path planning enhances the performance of a robotic platform. This approach ensures improved efficiency by pre-computing and

storing feasible velocities and accelerations, enabling access and execution of trajectories during real-time navigation. The platform gains enhanced responsiveness to dynamic environmental changes and unforeseen obstacles, adapting promptly without extensive recalculations. The library's consideration of physical constraints minimises unfeasible commands, promoting smoother movements and transitions. Additionally, resource conservation is realised as precomputed trajectories reduce the need for continuous online path planning computations, which is particularly advantageous for platforms with limited processing capabilities. Implementing an offline trajectories library comprehensively elevates the platform's efficiency, adaptability, safety, and resource utilisation.

The assessment of ATL's ability to navigate a platform with robustness involves a series of deployments on two mobile platforms. Through these deployments, we execute comprehensive experiments and trials to analyse how ATL performs under the condition of an outdoor unstructured terrain. These investigations are crucial for understanding the adaptability and efficacy of ATL in challenging environments.

Furthermore, we implement comparative analyses with other navigation techniques, specifically TEB (Time Elastic Band)(Rösmann et al., 2012), DWA (Dynamic Window Approach) (Fox et al., 1997) and the A* graph search method. This comparative assessment provides a balanced evaluation, allowing us to measure ATL's performance against previous methods.

By conducting side-by-side comparisons, we can conclude ATL's advantages in addressing a platform's physical limitations by implementing an offline trajectories library. Additionally, ATL demonstrated efficiency in managing the computational resources, leading to the reduction of execution planning time and enhanced performance on unstructured outdoor terrain for the target platform. We then demonstrate the effectiveness of ATL in real-world operation through experiments with a full-scale robot in an outdoor environment with uneven terrain containing several obstacles.

This thesis extends our work published as a conference paper at ACRA 2023, titled "Adaptive Trajectory Library Planner for Fast Outdoor Robot" (Nguyen et al., 2023). Nguyen Thanh Trung Le was the lead author of this publication.

1.5 Thesis Outline

This thesis is organised as follows:

Chapter 2 describes related work and focuses on exploring different types of previous and current path-planning techniques.

Chapter 3 focuses on the problem formulation and formally defines the problem addressed in the thesis.

Chapter 4 presents the Adaptive Trajectory Library (ATL) with details about our main contribution, a local planner for fast outdoor ground robot.

Chapter 5 analyses experiments, and discusses experiments for both simulation and hardware deployment.

Chapter 6 concludes and summarises the thesis and explores the improvements and future work.

Chapter 2

Related Work

In this chapter, we present a review of the literature related to autonomous path planning, aiming to provide a comprehensive understanding of the current state of this dynamic and rapidly evolving field. By examining and synthesising the relevant works, we seek to elucidate the landscape that defines our research focus in autonomous local path planning.

2.1 Traditional Path Planning

Traditional path planning methods have long been foundational in robotics, aiming to navigate robots from a start to a goal while avoiding obstacles. Among these methods, the Artificial Potential Field (APF) and Pure Pursuit approaches stand out for its simplicity and effectiveness. These traditional techniques continue to be relevant and are often integrated with modern algorithms to enhance path planning efficiency and adaptability in various robotic applications.

2.1.1 Artificial Potential Field

One of the earliest methods proposed was Artificial Potential Fields (Khatib, 1986), in which obstacles are modeled as repulsive fields and goal locations as attractive fields. APFs have long been fundamental in robotics and autonomous navigation. These fields are based

on the concept of potential energy, with the robot moving along paths of least potential. The attractive potential, generated by the target, pulls the robot toward its goal, while the repulsive potential, created by obstacles, pushes the robot away from collisions. Robots follow descending gradients in the resultant of these fields to reach the goal. The approach can practically be implemented for both global and local path planning.

APFs have been used for a variety of applications, including multiple types of robots, ranging from mobile robots (Melchiorre et al., 2023) (Park et al., 2001), Unmanned Aerial Vehicles (Khuswendi et al., 2011) (Cetin et al., 2009) (Shin and Kim, 2021), robot arm (Yuan et al., 2021), and multiple robots operation including the contribution of multiple types of robots (Warren, 1990) (Zhang et al., 2010).

This approach provides good obstacle avoidance (Park et al., 2001), but robots can get stuck in local minima, where they are caught up in a suboptimal configuration, unable to reach the global goal. A local minima represents a region where the robot is closely located to the obstacle or the obstacle position is in the middle of the goal and the robot, resulting in the gradient going to zero. This issue is particularly problematic in complex environments with narrow passages and multiple obstacles.

To resolve this problem, Koren and Borenstein (Koren and Borenstein, 1991) proposed one of the first solutions by using the Vector Field Histogram (VFH) method, which addressed local minima problems and improved obstacle avoidance. Additionally, the approach to combining APFs with other solutions to construct hybrid methods can highly improve the performance of APFs (Yuan et al., 2021) (Shin and Kim, 2021).

Overall, APF were one of the first solutions for a traditional path-planning problem, providing a comprehensive approach to utilising repulsive forces to find a considerable solution for a robot platform to reach its destinations. Yet, the algorithm can be easily affected and hardly overcome the local minima trap that results in a non-optimal outcome. Even though many approaches can largely improve the algorithms, various scenarios can produce different outcomes due to the algorithm's challenging real-world implementation.

2.1.2 Pure Pursuit

Another traditional path planning method is Pure Pursuit (Coulter, 1992b). The Pure Pursuit algorithm is also a foundational approach in robotic autonomous navigation. It maintains a consistent forward velocity for the robot, and its algorithm calculates the necessary angular velocities during the operating time, then guides the robot toward a specific point positioned at a predetermined distance in front of it. Pursuing a predefined point simplifies the control task, as the robot essentially follows a geometric path defined by the chosen point, helping to reduce the computational complexity associated with path planning.

Pure Pursuit's simplicity and ease of implementation make it an attractive choice for many robotic applications. From self-driving cars in urban environments (Peng et al., 2020) (Wang et al., 2017), compact racing cars (Sukhil and Behl, 2021) (Becker et al., 2023), skid-steered robot (Joglekar et al., 2022) and even an autonomous driverless electric crawler (Wu et al., 2023).

However, as technology advances and the complexity of robotic tasks increases, it becomes clear that Pure Pursuit has limitations, particularly in dynamic and cluttered environments. It primarily focuses on geometric considerations and does not account for the robot's dynamics or the presence of obstacles. This makes it less suitable for scenarios where the robot must navigate challenging terrain, avoid dynamic obstacles, or adhere to specific motion constraints.

Researchers have recognised these limitations and developed several extensions and variations of the Pure Pursuit algorithm. These modifications aim to enhance performance in real-world scenarios. For instance, Adaptive Pure Pursuit (Campbell, 2007) and Regulated Pure Pursuit (Macenski et al., 2023) introduce features that allow robots to adjust their linear velocity, reduce adjustments in direction, and respond more effectively to unexpected obstacles in their path. These enhancements make Pure Pursuit-based methods more versatile and adaptable, but they still remain constrained by the fundamental assumption that the robot's motion can be abstracted to a simple point-to-point pursuit without considering more dynamics of the platform.

In summary, Pure Pursuit is a valuable path-planning method for situations where geometric path following is sufficient and computational resources are limited. However, in scenarios where dynamic feasibility constraints, complex environments, or obstacle avoidance are primary concerns, alternative path planning techniques that consider the full dynamics and complexities of the robot's movement may be more appropriate for ensuring safety and successful navigation.

2.1.3 Conclusion

In conclusion, traditional path planning methods such as Artificial Potential Fields and Pure Pursuit offer effective approaches for navigating robots and autonomous systems. APF provides a straightforward yet powerful method by simulating forces that guide robots toward goals while avoiding obstacles. On the other hand, Pure Pursuit enables agile navigation by calculating steering commands based on a target point ahead of the robot, optimising trajectory tracking in dynamic environments. However, APF can sometimes struggle with local minima or getting stuck in complex obstacle configurations. Similarly, Pure Pursuit's reliance on a predefined path may lead to challenges in environments with unpredictable obstacles or dynamic changes.

2.2 Graph Seach

Graph search algorithms are fundamental tools in solving path planning problems across various domains, including robotics, computer science, and operations research. The algorithms, such as Dijkstra's (Dijkstra, 1959), A*(Hart et al., 1968), and D*(Stentz, 1994), are frequently utilised to determine feasible and efficient paths for robots to navigate. The following sections provide a detailed explanation of these graph search methods and their applications in robotic motion planning.

2.2.1 Dijkstra's Algorithm

Dijkstra's algorithm is a fundamental technique for finding the shortest path between nodes in a graph, often representing real-world networks like road systems. E.W. Dijkstra introduced this algorithm to the community in 1959, as documented in (Dijkstra, 1959). Dijkstra is widely used in many applications, especially for vehicle path planning (Zhu and Sun, 2021) (sang Liu et al., 2021) (Sun et al., 2021), stereo vision for road detection (Zhang et al., 2018) and many game applications (Bardi and López, 2015) (Huang and Yi, 2021). In the standard implementation of Dijkstra's algorithm, the approach seeks to determine the shortest possible path within a graph without explicit consideration of the practicality or feasibility of the solution.

Dijkstra's method is highly effective in solving problems related to identifying the shortest path from a source node to other nodes in a network. It achieves this by considering the associated cost to each node of the network to determine the path that minimises the overall distance or weight between the source node and every other node in the graph. This ensures that it can reliably identify a single shortest path from a source node to other nodes in the network.

However, in certain scenarios where the associated costs of nodes are prohibitively high or negative, the traditional Dijkstra algorithm may not be the most efficient choice (Sun et al., 2021) (Zhu and Sun, 2021). Therefore, many modified versions of Dijkstra have been introduced to resolve the problem, including the Improved Dijkstra's Shortest Path Algorithm (Shu-Xi, 2012) or the approach of improving data structure and restricted search area of the traditional Dijkstra (Fan and Shi, 2010).

2.2.2 A*

The A* Search method is known as one of the most efficient and widely adopted techniques for path-finding and graph traversals.

A* stands out as an informed search algorithm, categorised as a best-first search, uniquely tailored for weighted graphs. Starting from a chosen node, the task of A* is to seek the optimal path to move from the designated node to a specified goal node while limiting the overall cost of travel. These costs can be the travel distance, time expended, or other relevant criteria depending on the operation's objectives. A* maintains a queue of nodes to explore, prioritising nodes with the lowest cost. This allows A* to explore paths that are likely to be shorter, reducing the search space and improving efficiency compared to uninformed search algorithms.

A* shares similarities with Dijkstra's algorithm but introduces an additional element by integrating a cost-to-goal heuristic estimate into its cost function. This heuristic enhances the efficiency of the search process, typically resulting in faster exploration and pathfinding when compared to Dijkstra's algorithm.

While A* is a highly effective and widely used path-finding algorithm, it does have certain limitations in computational time and memory usage. It is not always the best technique for dynamic environments. When considering the application of A* to navigate through large graphs or grids, it's important to recognise that this approach may cause significant memory demands on the system. This problem occurs due to the necessity to maintain a record of explored nodes, potentially resulting in elevated memory usage and longer execution times to complete the search task. Furthermore, the performance of A* can be impacted by dynamic obstacles, as these elements can alter the cost values associated with nodes. Consequently, what might have initially been the shortest path may become a collision one due to these dynamic changes, resulting in the replanning of a new collision-free route.

To overcome some of the original algorithm's limitations, numerous enhancements and modifications have been proposed, as evidenced by research such as (Warren, 1993), (Ammar et al., 2016), Adaptive A* (Sun et al., 2008), Dynamic A* (Likhachev et al., 2005) (Souissi et al., 2013) and Incremental A* (Koenig and Likhachev, 2001). An interesting work by Ferguson (Ferguson and Stentz, 2007) introduces Field D*, an interpolation-based planning and replanning technique that helps ease the problems of the traditional version. Field D* is based on the concept of field planning. This technique is an extension of D*

and D* Lite, and it uses linear interpolation to construct low-cost pathways that minimise redundant turning effectively. The trajectories are optimal if one considers that linear interpolation is being used, and they are also quite effective in practical use. These improvements have led to more robust and versatile solutions, making A* a path planning and optimisation cornerstone.

2.2.3 Conclusion

The methods explored above can find the shortest distance to a goal location on a graph representation of the environment. However, constructing a graph is computationally expensive, thus reducing its suitability for robots operating at high speeds where decisions must be made quickly. Two-dimensional graphs are commonly used, but capturing constraints such as ensuring dynamic feasibility requires constructing the graph in higher state spaces. The search problem quickly becomes intractable as the A* can expand with a larger spatial scale or additional constraints.

The high computational cost of graph search methods can be reduced by minimising the graph size. This can be done by modelling the environment as a multi-resolution lattice state space, so the search can be performed at different scales to increase efficiency (Likhachev and Ferguson, 2009). However, this approach relies on heuristics, which can be hard to tune, and remains computationally expensive for the graphs necessary to operate in very large areas or with several dynamic feasibility constraints. Thus, graph and tree search-based methods can effectively benefit global path planning for Field Robotic operations. Another common approach is to use a separate global planner to find a path through a low-density graph, and assign the low-level navigation to a suitable local path planning algorithm such as those mentioned previously (Best et al., 2023).

2.3 Sampling-Based Algorithms

Sampling-based algorithms are a class of motion planning methods that address the complexities of high-dimensional spaces. These algorithms generate feasible paths by randomly sampling the configuration space and connecting these samples to form a roadmap or tree

structure. Popular approaches within this class include Probabilistic Roadmaps (PRM) and Rapidly-exploring Random Trees (RRT), which have been widely adopted due to their efficiency and ability to handle complex environments. This section explores the principles, implementation, and applications of sampling-based algorithms in robotic motion planning.

2.3.1 Probabilistic Roadmap

Probabilistic Roadmaps (PRM) have become a foundation technique in the field of robot motion planning, following their introduction by Kavraki et al (Kavraki et al., 1996) in the mid-1990s. This method has achieved significant recognition due to its effectiveness in navigating high-dimensional configuration spaces, a common challenge in robotics.

Firstly, the algorithm constructs a roadmap by randomly sampling points within the configuration space, representing feasible robot configurations. Collision-checking algorithms are used to checked and connect these nodes with edges, forming a graph that approximates the free space's connectivity. The pre-constructed roadmap is then used to find a path from a given start configuration to a goal configuration. This is done by connecting the start and goal positions to the nearest nodes in the roadmap and then performing a graph search, typically using algorithms like Dijkstra's or A*, to identify the shortest or most feasible path.

Over the years, PRM is widely used in a variety of applications, especially for robot arm manipulators (Hsu et al., 2005) (Jaillet and Simeon, 2004) and mobile robots (Le and Plaku, 2014) (Kumar and Chakravorty, 2012). Results from the experiments have demonstrated the robustness and effectiveness of PRMs in path planning tasks for high-dimensional robots.

Despite many advantages, PRM also have several disadvantages. One significant downside is their reliance on random sampling, which can lead to incomplete coverage of the configuration space, especially in highly constrained environments. Additionally, the quality

of the generated roadmap is heavily dependent on the number and distribution of sampled points, and achieving a dense and well-connected roadmap can be computationally expensive.

2.3.2 Rapidly Exploring Random Trees

Rapidly Exploring Random Trees (RRT) is widely regarded as one of the most influential robotics and motion planning algorithms. This algorithm, originally presented by Steven M. Lavalle (LaValle, 1998) (LaValle and Kuffner Jr, 2001), has improved how robots and autonomous systems navigate their environments. RRT ultilises an incremental sampling strategy in solving complex path-planning problems. The core idea behind RRT is to quickly explore the space of possible configurations by randomly sampling points and progressively expanding branches from the current configuration to create a tree-like structure. This tree, known as the RRT, aims to cover the feasible configurations within the environment efficiently. By tracing the branch with the closest node to the goal, RRT then constructs a path from the branch nodes to form a solution to reach the destination.

Additionally, RRT can be very efficient whether the configurations are in 2D or 3D configuration spaces (Kuffner and LaValle, 2000). Similar to other graph search techniques, RRT is practically deployed in many operations, including mobile robotic vehicle (Kuffner and LaValle, 2000) (Palmieri et al., 2016), Ackerman steering type (Peng et al., 2021), robotic arms (Grothe et al., 2022) to autonomous urban driving (Kuwata et al., 2009).

While RRTs have many advantages, they also have certain limitations, which include the algorithm's reliance on random sampling. RRTs are inherently probabilistic and can produce different results for the same problem in multiple runs. While the random approach can avoid local minima, one disadvantage is this randomness can make them less predictable and sometimes challenging to analyse. Moreover, RRTs tend to explore areas of the configuration space where the tree has not expanded much, which can result in biased sampling towards unexplored regions. This bias can lead to suboptimal or inefficient paths, especially in situations where there are narrow passages or tight constraints. RRT*, first introduced by Karaman (Karaman and Frazzoli, 2010) (Karaman and Frazzoli, 2011),

has significantly improved the traditional method to resolve these limitations. The algorithm tracks the distance traveled by each node relative to its parent using a cost function. When it identifies the nearest node within the graph, it examines nearby nodes within a set radius. If a node with a lower cost is found, it replaces the previous one.

2.3.3 Conclusion

In conclusion, both Rapidly-exploring Random Trees and Probabilistic Roadmaps represent significant advancements in sampling-based algorithms for robotic motion planning. RRT performs well in rapid exploration of configuration spaces, making them suitable for dynamic environments and real-time applications. PRM, on the other hand, provide a structured approach to path planning with probabilistic completeness, offering robust solutions in complex scenarios. However, graph search methods can involve extensive computational processing, leading to higher latency and potentially reduced responsiveness compared to other approaches like the Dynamic Window Approach or Time Elastic Band.

2.4 Kinodynamic Planning

To achieve effective motion planning, it's crucial to align with the physical dynamics of the platform. This section explains how kinodynamic planning methods address dynamic challenges by ensuring that the planned trajectories satisfy the target platform's velocity, acceleration, and torque constraints. Additionally, this section details optimisation-based methods and trajectory libraries, which are frequently deployed to navigate these dynamic considerations effectively in path planning.

2.4.1 Optimisation Method

Recent advancements in local path planning with optimisation-based techniques have become prominent. This section details the Elastic Band (EB) and its time-dependent variant, the Time Elastic Band (TEB), have proven effective in dynamically adjusting paths

to navigate complex environments precisely and efficientl

The Elastic Band, introduced in (Quinlan and Khatib, 1993), has provided an innovative approach to address the complexities of local path planning. This method leverages the power of least-squares optimisation to effectively deform a predefined commanded path to avoid environmental obstacles while addressing the robot's dynamic constraints. By employing optimisation, the Elastic Band optimises the path, modifying it in real-time to navigate around potential collisions and obstacles, ensuring the robot can operate safely and efficiently.

The Time Elastic Band, an evolution of the Elastic Band paradigm, incorporates the temporal dimension into the path planning process (Rösmann et al., 2012; Rösmann et al., 2017). This extension enables the method to account for time-varying obstacles and evolving dynamic constraints, providing more versatile and adaptable local path planning. The Time Elastic Band relies on optimisation techniques to handle the path's deformation while considering both spatial and temporal aspects of the navigation problem. This added temporal awareness is particularly useful in scenarios where obstacles or robot dynamics change over time.

These optimisation-based methods adapt and optimise paths in real-time, while incorporating dynamic feasibility and obstacle avoidance. This has made them valuable tools for autonomous systems, catering to a wide range of robotic platforms and applications. Whether it be indoor navigation (Wu et al., 2021) (Smith et al., 2020), multi-robot navigation (Chung et al., 2022), robot arm (Magyar et al., 2019), or complex dynamic environments, optimisation-based approaches continue to push the boundaries of what is achievable in the realm of autonomous navigation, ensuring safe and efficient robot movement in various challenging scenarios.

Optimisation kinodynamic planning solutions effectively address the challenges of adapting planners to a target platform's physical capabilities and design. These approaches present a range of robust and efficient techniques designed to enhance the planner's ability to navigate complex environments while considering the dynamic constraints of the platform. This includes optimisation approaches like Trajectory Optimisation (Schulman

et al., 2013), Stochastic trajectory optimisation for motion planning (STOMP) (Kalakrishnan et al., 2011), Guaranteed Sequential Trajectory Optimisation (GUSTO) (Bonalli et al., 2019), Covariant Hamiltonian optimisation for motion planning (CHOMP) (Zucker et al., 2013), to state lattice-based search (Pivtoraiko and Kelly, 2005) and Discontinuity-Bounded Search (Hönig et al., 2022). Each method helps the planner adapt to the target platform's physical capabilities and design, effectively solving the core challenge of kino-dynamic planning.

However, the optimisation solution quality is subject to trade-off against computation time and, thus, is unsuitable for high-speed operations. In such dynamic environments, longer computational processing times can potentially lead to unforeseen collisions or accidents as the robot navigates at high speeds.

2.4.2 Dynamic Window Approach

The Dynamic Window Approach (DWA) is a versatile method in the field of mobile robotics, designed to optimise path planning and collision avoidance for autonomous agents. Introduced by Fox et al (Fox et al., 1997), the DWA algorithm has proven to be a powerful tool for enhancing the agility and safety of mobile robots in dynamic and cluttered environments. Here, the potential paths resulting from a range of linear and angular velocities are simulated over a short time period using the differentially steered drive kinematic equation (Lucas, 2001). A suitable instantaneous trajectory is chosen by assigning a cost to each that trades off between the distance to obstacles and how closely the given path is followed.

From indoor robots (Choi et al., 2012) (Patel et al., 2021) (Yasuda et al., 2023), UAVs (Qin et al., 2023) (Wang et al., 2023b) (Wang et al., 2023a), ship planning (Lu et al., 2022) to underwater vehicles (Li and Zhang, 2023) (Jian et al., 2020), DWA is popularly used in the field of autonomous navigation, widely implemented as a local path planning for platforms that operate within different environments.

DWA has been found to perform well at calculating dynamically feasible motion plans relatively quickly (Yuan et al., 2022) (Cybulski et al., 2019), but the online computation

of the paths takes time and can lead to poor or unpredictable path selections when run at high frequencies required for high-speed robots. Moreover, in areas with dense obstacles, DWA tends to fail or to favor going around the outside of the obstacle region, resulting in an increased total traversal distance (Qin et al., 2023). Furthermore, DWA faces challenges when navigating around C-shaped obstacles, often leading to failures in the objective cost function and, consequently, an inability to compute a valid path (Yan et al., 2022).

In conclusion, DWA introduces one of the most popular techniques in addressing a path-planning problem, showcasing how a simple yet elegant concept can lead to an effective solution. However, since the online computation of DWA takes time and can result in poor or unpredictable selections for high-speed robots, there are limitations to consider using DWA. Thus, it is necessary to develop more robust and efficient techniques to produce better performance.

2.4.3 Offline Trajectory Library

To increase the responsiveness of motion planners, researchers have recently explored using a library of dynamically feasible trajectories generated offline. Such libraries of collision-free, dynamically feasible trajectories have been used to efficiently calculate paths for robotic manipulators to achieve repetitive tasks, such as removing weeds from fields (Lee et al., 2014), Pruning Trees (You et al., 2020), Advanced Mobility Quadrupedal Robots (Bjelonic et al., 2022) and UAVs (Viswanathan et al., 2020).

An offline trajectory library provides a valuable solution to address one of the most significant challenges associated with online velocity sampling methods such as DWA or TEB. These online methods involve repeatedly generating and evaluating trajectories during each iteration of planning and control, which can introduce a substantial delay during runtime. This delay, if not managed effectively, can potentially have a negative impact on the overall performance of the robotic platform. On the other hand, the offline trajectory library offers a precomputed set of trajectories that are strategically generated and stored before runtime. These trajectories are typically designed to cover various scenarios and environmental conditions the robot may encounter during its mission. When the robot is

in operation, it can select and execute trajectories from this library in real-time based on its current state and the surrounding environment.

A similar approach has also been applied to quadrotor aerial robots (Best et al., 2023). In this context, the goal is to navigate quadrotor drones in complex environments efficiently. When these drones must follow a specified path generated by an upstream planner, a path-planning algorithm is crucial in ensuring their safe and optimal operation. The process begins with the global planner providing the desired path for the quadrotor. This path is a series of waypoints that define the desired trajectory the drone should follow. However, simply instructing the drone to follow these waypoints in a straight line may not be feasible, as it could lead to collisions or require maneuvers that the drone is physically incapable of performing.

To address these challenges, a path-planning algorithm is employed. This algorithm's primary task is to select the closest collision-free trajectory from a precomputed library. This library contains a variety of trajectories, each tailored to specific scenarios and environmental conditions. The trajectory choice is made based on several factors, including the current state of the drone and the characteristics of the environment it is navigating through. One key aspect of this approach is the consideration of dynamic feasibility. The selected trajectory must be collision-free and compatible with the drone's current state, including its position, velocity, and orientation.

In summary, the approach applied to quadrotor aerial robots involves a combination of path planning and trajectory selection. It enables drones to follow a desired path while considering environmental constraints and the drone's current state, ultimately ensuring safe and efficient navigation in complex and dynamic surroundings. This thesis explores how this principle could be generalised to a ground robot moving at high speeds by demonstrating its performance against other solutions throughout practical simulation and hardware experiments. By showcasing the outcomes of these trials, we underscore the need for implementing ATL in outdoor robots navigating through unstructured terrain.

Planner	Type	Limitation
Pure Pursuit	Controller	Does not consider the platform's dynamics
Artificial Potential Fields	Controller	Gets stuck in local minima
A*, D* and Dijkstra	Graph Search	Requires expensive computational resources
RRT	Tree-based	Requires expensive computational resources
DWA	Predictive	Tends to oscillate near obstacles
EB and TEB	Optimisation	Not suitable for fast planning
Offline Trajectory Library	Trajectory Library	Only used in UAVs and Robot Arms
Adaptive Trajectory Library	Trajectory Library	Requires high-quality obstacle map

TABLE 2.1: Limitations of path planners

2.4.4 Conclusion

In conclusion, kinodynamic planning represents a critical advancement in robotic motion planning by integrating both kinematics and dynamics considerations. This approach enables robots to navigate through complex environments while considering to both motion constraints and dynamic feasibility. By simulating and optimising trajectories that account for both robot dynamics and environmental constraints, kinodynamic planners enhance path efficiency and safety in diverse applications, from industrial automation to autonomous vehicles. Continued research and development in kinodynamic planning promises further advancements in robot autonomy and performance across various real-world scenarios.

2.5 Conclusion

Many techniques and solutions have emerged as technology advances to address diverse path-planning challenges across various conditions and constraints. From approaches APFs, to graph and tree search algorithms such as A*, D*, and RRT, as well as predictive strategies like the DWA and optimisation-based methods like EB and TEB, the field of autonomous navigation for robots has experienced rapid and expansive growth. Nevertheless, each method operates within its own set of limitations, as shown in Table 2.1

The table has outlined the limit of the performance of preceding planners. However, given that implementing the Offline Trajectory Library has been exclusive to UAVs and Robot Arm planning, we are excited to explore and analyse the potentially profound impact this method could display in outdoor ground robots.

In conclusion, traditional methods (Pure Pursuit and Artificial Potential Field) are unsuitable for modern path-planning problems. Modern problems are more challenging in different aspects, from the structure of the operating terrain to the variations of observation sensors and the information they can provide, which makes these traditional methods unable to the more complex scenarios considered by contemporary researchers.

On the other hand, graph search and tree search methods (RRT, PRM, and A*, D*) are more suitable to be considered as global path planning for an operation. Even though they can provide an effective and robust path-planning solution, these methods rely on the computing system's performance, as their crux algorithm demands lots of computational time to identify the shorted route to the destination.

Nevertheless, although DWA and TEB are famous for local path planning, these techniques have their own weakness. For DWA, the solution can be very efficient and predictive within a limited obstacle environment, however, DWA frequently exaggerates turning command while maintaining a low linear velocity when the target platform is close or located near a dense obstacle area. As we observed this behaviour from experiments, DWA tends to do this to navigate the robot by turning away from the obstacle, but, depending on the dynamics of the platform, this control action might not be feasible, resulting in the platform's immobilisation near the obstacle.

TEB is an optimisation-based method that can quickly adapt to the conditions of the surrounding environments. TEB is highly effective and widely used for applications where the operating area contains multiple obstacles, even the dynamic type. However, TEB includes a set of customised parameters that define the planner's behaviour. These parameters demand a significant tuning time with efforts to increase the performance of TEB, otherwise, the performance can be very poor.

Finally, offline trajectory library methods have recently been implemented as solvers for path planning. The method has proved to significantly improve the performance of the target's platform, where the robot can execute a robust and effective movement to reach certain destinations while ensuring its dynamic capabilities. Offline library methods, however, have not been utilised for mobile robotic platforms to investigate their compatibility and effectiveness. Thus, in this thesis, we propose an approach that combines an offline trajectory library for field robotic path planning called the Adaptive Trajectory Library to identify and analyse the performance of this method on the target platform.

Chapter 3

Problem Formulation

The fundamental challenge of our analysis revolves around efficiently guiding a platform through a specified trajectory. This trajectory, encapsulated as a series of poses from a distinct orientation and destination, is presented to the planner for processing. This task divides our focus into two critical aspects of the planning problems.

3.1 Efficient Path Planning

Our main goal is to identify the platform's most efficient and effective path. The planner is tasked with delivering a trajectory-compliant solution and optimising the path to minimise the temporal and resource footprint. By selecting the one that attains the destination in the shortest possible time with the least resource consumption, the planner fulfils its role in contributing to an operation that is not just trajectory-compliant but also resource-optimised.

Nevertheless, incorporating kinodynamic planning enhances this approach by considering both the kinematics and dynamics of the platform. This involves accounting for the platform's velocity, acceleration, and forces acting on it, ensuring the path is feasible regarding spatial constraints and adapts to its physical limitations. By integrating kinodynamic constraints, the planner can provide a more robust and realistic solution, ensuring the chosen

path is executable given the platform's dynamic capabilities, ultimately achieving optimal performance in both time and resource efficiency.

3.2 Problem Formulation

The problem considered in this thesis is formally defined as follows

Considering a robot described by an N-dimensional state $\mathbf{x}_t$ (e.g., position and orientation) and an M-dimensional control action $\mathbf{u}_t$ (e.g., translational and rotational velocity command) at each time t. Feasible states and control actions are limited to subsets $\mathbb{X} \subset \mathbb{R}^N$ and $\mathbb{U} \subset \mathbb{R}^M$ respectively. The state $\mathbf{x}_t$ and the control actions $\mathbf{u}_t$ are governed by a dynamic model $\mathbf{f}$:

$$\dot{\mathbf{x}}_t = \mathbf{f}(\mathbf{x}_t, \mathbf{u}_t). \tag{3.1}$$

Importantly, in addition to the limitations modelled by feasible state- and control-spaces $\mathbb{X}$ and $\mathbb{U}$, the robot is subject to a dynamic feasibility constraint:

$$\mathbf{u}_t \in \mathcal{D}(\mathbf{x}_t, \dot{\mathbf{x}}_t). \tag{3.2}$$

This can model a variety of constraints ranging from simple ones such as turning radius and acceleration limits in a bicycle model, to complex ones such as slip prevention for off-road operation. In addition to dynamic constraints, the robot operates in an environment with obstacles to avoid, denoted by $\mathcal{O} \subset \mathbb{X}$.

We are given a desired path $\mathbf{X}^d = \mathbf{x}_t^d \dots \mathbf{x}_{t+T}^d$ from a higher-level planner that may not be collision-free due to factors such as a slow update rate. The primary objective of this paper is to develop a strategy for planning a minimum-time path, denoted as $\mathbf{X}$, which closely aligns to a predefined desired path $\mathbf{X}^d$.

The output trajectory must also visit all points in the desired trajectory in the specified order, as indicated by the constraint in (3.3) (line 3). Each point in $\mathbf{X}^d$ is considered visited if there exists at least one point in $\mathbf{X}$ where the Euclidean distance to the point

is less than a given threshold. This verification ensures that the output trajectory $\mathbf{X}$ adequately covers all points in the desired trajectory $\mathbf{X}^d$.

In addition to this proximity requirement, the planned path must satisfy several critical constraints. Firstly, the path must avoid any potential collisions with environmental obstacles. Secondly, the path must consider the dynamic feasibility constraints, which involve ensuring that the motion along the path is within the capabilities of the vehicle or system in question, considering factors such as maximum velocity, acceleration, and other dynamic limits.

$$\begin{aligned} \min_{\mathbf{X}} \quad & \text{travel time} \\ \text{s.t.} \quad & \text{dynamics (3.1)}, \\ & \text{dynamic feasibility (3.2)}, \\ & \mathbf{X} \cap \mathcal{O} = \emptyset \\ & \mathbf{X} \text{ visits } \mathbf{X}^d \end{aligned} \tag{3.3}$$

The main challenge in solving (3.3) is the dynamic feasibility constraint (3.2). The set of available control actions depends on the state, which introduces significant complications because states are also dependent on control actions. Despite such co-dependence, the planner must be robust against inaccuracies or even a lack of a model of the dynamic feasibility constraint (3.2), as may be the case for practical robots.

Chapter 4

Adaptive Trajectory Planner

This chapter introduces the Adaptive Trajectory Planner (ATL) as an approximate solver for Equation (3.3). ATL addresses three key challenges: kinodynamic planning, optimal utilisation of computational resources, and strategic planning for ground robots operating in unstructured outdoor environments. ATL implements an offline library that stores offline trajectories, each comprising a dynamically feasible pair of linear and angular velocities with a simulated trajectory generated from the pair of velocities. Each trajectory is associated with a simulated trajectory generated using a forward kinetic algorithm based on the corresponding velocity tuple. Through a comparative analysis of these offline trajectories, ATL identifies the optimal one and extracts the velocity pair to guide the robot's controller along the commanded path. The following sections dig into the specifics of each stage in the ATL process.

4.1 Overview

ATL stands as an innovative and sophisticated local planner solution for outdoor ground mobile robots. The methodological foundation of this approach is explained in Section 4.2, where an offline library is generated, consisting of a diverse range of offline trajectories.

These offline trajectories are generated given a forward kinematic equation for ground mobile robots, which is explained in the same section. This offline library serves as the backbone for the planner's operation, enabling the identification of the optimal collision-free path, as explained in Section 4.3.2. The uniqueness of ATL lies in its dynamic trajectory filtering mechanism, detailed in Section 4.3.1, ensuring real-time dynamic feasibility by aligning with the robot's current state. Section 4.4 further delves into specific considerations essential for implementing the planner on a robotic platform. From addressing hardware constraints to optimising computational efficiency, these considerations provide informative insights for new operators aiming to deploy ATL in real-world scenarios.

By integrating an offline trajectory library into a local path planning solution for outdoor terrain operations, ATL appears not only as a theoretical framework but as a practical and adaptive solution capable of navigating the complexities inherent in the mobility of ground robots.

4.2 Offline Library Construction

This marks the initial phase of ATL, serving as an offline step, in which it won't be reinitialised for the online planning phase during the operation.

The initial step involves capturing a set of velocities, a combination of both linear and angular velocities. This range spans from the minimum to maximum values for angular velocities, aligning with the physical capabilities of the target platform. To obtain accurate and practical data, these selective values are recorded through real-time experiments involving the physical robot. During these experiments, operators document the actual velocities executed by the platform, ensuring that the recorded values authentically represent the platform's dynamic performance. This approach effectively addresses the dynamic constraints inherent to the platform, recognising that physical limitations can influence the actual performance of the robot. All of the recorded velocities are stored in a control action set called $\mathbf{U}^C$.

To ensure the dynamic feasibility of the platform, we first construct a container that holds a set of trajectories, which get called at the first stage of the planner. We define the

offline library $\mathbf{L}$ as a set of tuple T, where each $T = (\mathbf{X}, \mathbf{U})$ comprising a trajectory $\mathbf{X}$, and a pair of linear and angular velocities $\mathbf{U}$ in the $\mathbf{U}^C$. $\mathbf{X}$ is a trajectory that consist of poses denoted as $\mathbf{x}_i$. These poses are generated through Algorithm (1) with the associated control action $\mathbf{U}$, where each control action is a combination of feasible linear velocity $\mathbf{v}_x$ and feasible angular velocity ω_z. $\mathbf{x}_i$ is a combination of position in ($\mathbf{x}$, $\mathbf{y}$) axes and the heading orientation θ, with $\mathbf{i}$ indicate the index of the pose within the trajectory $\mathbf{X}$. The trajectories are in the egocentric frame of the robot so that they are independent of its pose.

The merit of ATL is that such a model does not need to be explicitly constructed. Rather, control actions may be tested at different states in simulations and hardware trials with diverse configurations and environmental conditions. Similarly, a first-principles model may be adjusted through experimentation.

The dynamic models used to construct the offline library in Algorithm 1 can be changed to match the specific dynamics of various target robotic devices, such as quadcopters or Ackermann steering cars. This flexibility allows users to create an adaptable library customised to their specific robots' unique characteristics and performance requirements. By adjusting the models to reflect the real-world dynamics of different robotic platforms, users can achieve more accurate estimation for the offline trajectories to optimise performance. This adaptability is crucial for developing robust robotic systems that handle various tasks and environments. For instance, Section 5.3 presents examples of the offline library for two different steering capabilities of the two mobile robots.

To determine this set of state trajectories, a standard dynamic algorithm can be used to construct the offline library $\mathbf{L}$:

4.3 Online Trajectory Selection

4.3.1 Online Trajectories Filtering

Before advancing to the final matching phase, a critical preparatory step involves the filtering of offline trajectories. This process serves to eliminate infeasible paths that are

Algorithm 1 Offline Library Construction

Inputs: Control Action Set $\mathbf{U}^C$
Parameters: Simulation Time $\mathbf{t}$, Time Step $\mathbf{d}^t$
Outputs: Offline Library $\mathbf{L}$

▷ The first pose of the trajectory is the origin of the platform
$\mathbf{x}_0 = (0, 0, 0)$

▷ Generate an estimated trajectory for each control action in the set
for each control action $\mathbf{U} \in \mathbf{U}^C$ **do**

▷ Estimate the next potential pose of the offline trajectory
  for $\mathtt{i} = 1, 2, ..., \mathbf{t}/\mathbf{d}^t$ **do**
    $\mathbf{x}_i(x) \leftarrow \mathbf{x}_{i-1}(x) + (\mathbf{U}(\mathbf{v}_x) \cdot \cos(\theta) - \mathbf{U}(\omega_z \cdot \sin(\theta)) \cdot \mathbf{d}^t$
    $\mathbf{x}_i(y) \leftarrow \mathbf{x}_{i-1}(y) + (\mathbf{U}(\mathbf{v}_x) \cdot \sin(\theta) + \mathbf{U}(\omega_z \cdot \cos(\theta)) \cdot \mathbf{d}^t$
    $\mathbf{x}_i(\theta) \leftarrow \mathbf{x}_{i-1}(\theta) + \mathbf{U}(\omega_z) \cdot \mathbf{d}^t$

▷ Add this tuple to the library
  $\mathbf{L} \leftarrow \mathbf{L} \cup \{[(\mathbf{x}_0, \mathbf{x}_1, ...\mathbf{x}_{\mathbf{t}/\mathbf{d}^t}), \mathbf{U}]\}$
return L

not suited to the present state of the system, ensuring the requirement of (3.2), where the platform is limited by the dynamic feasibility constraint in control action at certain states.

Considering a scenario where a platform is in motion with a linear velocity of $\mathbf{v}_x$ meters per second and an angular velocity of ω_z radians per second. To accomplish this, *Dynamic thresholds* are employed, setting the limits at $\mathbf{k}^x$ meter per second for linear velocity and $\mathbf{k}^w$ radians per second for angular velocity. This approach allows the planner to systematically go through many potential paths and skip those whose profiles fall outside the prescribed boundaries.

We establish dynamic thresholds, denoted as $\mathbf{k}^x$ and $\mathbf{k}^w$. Each threshold is employed independently to define upper and lower bounds for velocities. These bounds are calculated by adding and subtracting the dynamic thresholds from the current linear and angular velocities, resulting in four distinct values, denoted as $\mathbf{u}^v$, $\mathbf{l}^v$, $\mathbf{u}^w$, and $\mathbf{l}^w$. Subsequently, we assess whether the current control action $[\mathbf{v}, \mathbf{w}]$ with linear velocity $\mathbf{v}$ and angular velocity $\mathbf{w}$, associated with the corresponding $\mathbf{X}$, falls within these bounds. This evaluation allows us to determine the feasibility of the offline trajectory concerning the robot's current state.

By integrating this trajectory filtration process, the planner ensures that only the viable options are considered for the subsequent planning stages, optimising the overall efficacy

Algorithm 2 Online Trajectories Filtering

Inputs: Current Library $\mathbf{L}$, Current Linear Velocity $\mathbf{v}_x$, Current Angular Velocity ω_z
Parameters: Dynamic thresholds $\mathbf{k}^x, \mathbf{k}^w$
Outputs: Filtered Library $\mathbf{L}^D$

▷ Find Upper and Lower Boundaries for Linear and Angular Velocites
$\mathbf{u}^v \leftarrow \mathbf{v}_x + \mathbf{k}^x$
$\mathbf{l}^v \leftarrow \mathbf{v}_x - \mathbf{k}^x$
$\mathbf{u}^w \leftarrow \omega_z + \mathbf{k}^w$
$\mathbf{l}^w \leftarrow \omega_z - \mathbf{k}^w$

▷ Filter feasible tuple from the offline library
for each $[\mathbf{v}, \mathbf{w}] \in L$ **do**

▷ Checking Linear Velocity
  if $\mathbf{v} \leq \mathbf{u}^v$ & $\mathbf{v} \geq \mathbf{l}^v$ **then**

▷ Checking Angular Velocity
    if $\mathbf{w} \leq \mathbf{u}^w$ & $\mathbf{w} \geq \mathbf{l}^w$ **then**

▷ If feasible, add this tuple to the collection
      $\mathbf{L}^D \leftarrow \mathbf{L}^D \cup \{[\mathbf{v}, \mathbf{w}]\}$
return $\mathbf{L}^D$

of the system's navigation strategy, making the planner adapt to the requirement of a kinodynamic method, where physical characteristics of the platform are required to be accounted.

4.3.2 Online Trajectory Matching

The online planning phase of ATL starts with the online filtering process as mentioned in the previous Section 4.3.1. Using the Dynamic Offline Library L^D constructed from the filtering phase, the online planning stage identifies the best collision-free trajectory $\mathbf{X} = \mathbf{x}_1 \dots \mathbf{x}_T$ over a short time horizon T that is both dynamically feasible and the closest to the next goal $\mathbf{x}^d \in \mathbf{X}^d$. Formally, the planning algorithm solves the following problem at each time t as mentioned previously in Equation 3.3

The process for solving Equation 3.3 using the library T is outlined in Algorithm 3, and illustrated in Figure 1.3. The online planning algorithm takes as input the current state $\mathbf{x}_t$, the desired path $\mathbf{X}^d$, and any obstacles $\mathcal{O}$, and outputs the best control action $\mathbf{u}_t^\star$ for the current time.

Algorithm 3 ATL Planner at time t

Inputs: Current state $\mathbf{x}_t$, desired path $\mathbf{X}^d$, obstacles $\mathcal{O}$
Outputs: Best control action $\mathbf{U}_t^\star$.

▷ Filter out infeasible trajectories
1: $\mathbf{L}^D = \text{OnlineTrajectoriesFiltering}(\mathbf{x}_t(\mathbf{v}_x), \mathbf{x}_t(\omega_z))$

▷ Initialise estimated best cost and control action
2: $J^\star$, $\mathbf{U}_t^\star \leftarrow \infty, \text{NULL}$

▷ Iterate over collections
3: **for** each tuple $T = (\mathbf{X}, \mathbf{U})$ in Filtered Library $\mathbf{L}^D$ **do**

▷ Check if trajectory is in collision
4: **if** $T_i(\mathbf{X}) \cap \mathcal{O} = \text{NULL}$ **then**

▷ Compute cost J of trajectory. If better, save.
5: **if** $J(T_i(\mathbf{X}), \mathbf{x}^d) \leq J^\star$ **then**
6: $J^\star$, $\mathbf{U}_t^\star \leftarrow J(T_i(\mathbf{X}), \mathbf{x}^d)$, $T_i(\mathbf{U})$
7: **return** $\mathbf{U}_t^\star$

The algorithm maintains an estimate of the current best cost $J^\star$ and current best control action $\mathbf{u}_t^\star$, which are initialised as ∞ and a null action respectively (line 2). We then loop over all trajectories in the Filtered Library (line 3) while skipping collision trajectories (line 4).

The cost function J in Algorithm 3 is determined by calculating the Euclidean distance between the next pose of the desired trajectory with the end pose of a trajectory within the Filtered Library $\mathbf{L}^D$. The algorithm iterates over every end pose in $\mathbf{L}^D$ to compute each J, aiming to identify the trajectory that has the minimum J value, denoted as $J^\star$. Throughout this process, only one pose from the desired trajectory is considered in each calculation, progressing until the target platform reaches this specific pose.

Computation time is reduced owing to such skipping because the cost function is only evaluated for dynamically feasible and collision-free trajectories, and collision-checking is only performed if dynamically feasible. During collision checking of a single trajectory, points along it are sampled in turn from that which is closest to the robot. As soon as a point that coincides with an obstacle is found, the trajectory is discarded without evaluating any additional points, further reducing computation time. The estimate of best cost and control action is updated if the current cost $J(T_i(\mathbf{X}), \mathbf{x}^d)$ is lower than the

estimate $J^\star$ (Alg. 3, line 5). After exhausting the trajectories in all collections, the best control action $\mathbf{u}_t^\star$ is returned.

If all trajectories fail collision checking, the algorithm may return a NULL control action. This implies that all dynamically feasible trajectories, given the current state can lead to a collision. In this case, a safety behaviour must be triggered. For ground robots, for example, we found stopping at the current position with zero velocity to be a suitable safety behaviour: this is not only because it is safe to do so, but also because the dynamically feasible trajectories when the robot has zero velocity are shorter in length than when moving at speed, and hence more amenable to escaping the obstacles. Meanwhile, the dynamic feasibility check in Algorithm 3, line 1 does not cause a NULL return as long as the feasible sets span the entire state space (i.e. $\bigcup_C \mathcal{F}^C = \mathbb{X}$).

When the matching process is executed, there are certainly be some discontinuities since the robot's current state and the library's state won't match exactly. These problems can be managed under certain assumptions:

- **Continuous Control Inputs:** Assume that the control inputs associated with the chosen trajectories in the library are sent continuously during the operation. This continuity helps smoothly transition between different trajectories without unexpected changes in control.
- **Feedback Control:** Assume the feedback control mechanism can adjust the robot's motion based on real-time sensor data. This allows the robot to correct deviations from the planned trajectory and maintain smooth, continuous motion.
- **Consistency in Robot Dynamics:** Assume that the robot's dynamics (including mass, inertia, wheel properties, etc.) are consistent across all trajectories in the library. This consistency ensures that the control inputs computed from the library trajectories are applicable and effective for the robot's physical characteristics.

4.4 Considerations for Implementation

This section explains how ATL can be integrated into a robotic platform. Each subsection presents different aspects of ATL, from the integration to ROS (Robotic Operating System) in Section 4.4.1, the offline trajectories reference frame in Section 4.4.2, obstacle detection in Section 4.4.3 to the configuration parameters that generate the trajectories in Section 4.4.4. This explanation covers diverse operational contexts, providing informative details to implement ATL into an operation or experiment.

4.4.1 ROS Integration

To evaluate the implementation of ATL for a standard path planning problem, we have chosen the approach of integrating ATL as a plugin within the ROS (Robot Operating System) base local planner package. ROS is an open-source framework designed to facilitate software development for robots. ROS is not a traditional operating system but rather a collection of tools and libraries, aimed at simplifying the complexity of robot software development. Integrating ATL as a plugin within the original package benefits the compatibility with the ROS navigation stack. This expanded connectivity enables the planner to access the full advantages of ROS, which links a diverse set of packages to contribute to a comprehensive path-planning pipeline. With ATL as a plugin, the planner becomes part of the ROS system. ROS provides a standardised and modular framework for robotics, offering access to various packages and libraries. This integration allows the planner to leverage the extensive capabilities of ROS while focusing on path-planning enhancements. The ROS navigation stack comprises multiple packages designed to address different aspects of navigation, including obstacle detection and avoidance, localisation, and mapping. ATL can directly interface with these packages to enhance the entire path-planning pipeline by becoming part of this ecosystem.

The modularity and standardised interfaces of ROS simplify the process of incorporating ATL into various robotic applications. It allows for the efficient exchange of different path-planning plugins and the extension of functionality with minimal integration effort.

4.4.2 Reference Frame

The trajectories in the offline library $\mathbf{L}$ are stored in the egocentric frame of the robot to obviate the need for simulating control actions at runtime. As the global static frame is typically specify the desired path $\mathbf{x}^d$ from the higher-level planner, thus, if each trajectory in the offline library were stored in the global static frame, converting them individually for each cost function evaluation at runtime would incur a significant computational cost.

4.4.3 Obstacle Detection and Avoidance

One of the notable features of the ROS navigation stack is the Costmap2D (Marder-Eppstein et al., 2023), a package designed for obstacle detection and avoidance. ATL, now integrated into this ecosystem, can easily interact with Costmap2D, using its real-time information on obstacle positions and safety parameters to adapt and optimize trajectory planning dynamically. For practical implementation, we recommend maintaining an independent obstacle representation from that of the higher-level planners. ATL plans over a smaller spatial scale and at a much greater frequency than typical high-level planners. Two criteria are important for selecting a suitable obstacle representation, which is updated time and availability of egocentric operation. In other words, it should be possible to update the obstacle representation quickly, and the obstacle representation should naturally support the egocentric frame as its reference.

We found that 2D occupancy grid (Thrun, 2003) implemented in `costmap_2d` (Marder-Eppstein et al., 2023) fits these two criteria well. It provides an intuitive and robust approach to obstacle detection by employing a 2D occupancy grid, effectively representing the robot's surroundings based on data gathered from its sensory observations. Each cell within this grid assigns a value reflecting the cost associated with navigating through that region. Closeness to an obstacle results in higher cost values, indicating potential risks to the platform. By leveraging these grid values within the cost function of ATL, trajectories with higher costs are systematically avoided, helping to reduce the computational duration in determining the most optimal path for the robot.

4.4.4 Parameters Setup

The package comes with two distinct configuration files, which directly affect the performance of ATL during its operational phase. These two play major roles in constructing the customised estimated trajectories for the target platform, ensuring that these trajectories align with the dynamic capabilities of the robot. The two files are:

- **Trajectory Config**: This configuration file serves to specify pairs of both linear and angular velocities, extracted from recorded data obtained from the platform. The recorded velocities are used to estimate the process of offline trajectories.
- **Library Config**: This configuration file focuss on the characteristics of the offline trajectories. There are five parameters and they are listed below:
  - *sim_time*: define how long in seconds should ATL estimate the offline trajectories
  - *sim_granularity*: define the step size in meters for each estimation poses of a single offline trajectory
  - *max_step*: define the limit for the number of poses in each offline trajectory
  - $robot\_frame$: define the egocentric frame of the robot, the offline trajectories are constructed and stored within this reference frame
  - *is_holonomic*: define whether the target platform is a holonomic robot or not. The holonomic robot has additional horizontal movement estimation for the offline trajectories.

Additionally, ATL is equipped with the Dynamic Reconfigure features. Dynamic Reconfigure, a ROS package (Carroll, 2023), enables users to adjust parameter values during the operational phase dynamically. Dynamic Reconfigure allows users to modify ATL's parameters that influence the tolerances towards the goal or the dynamic threshold for the filtering phase. Notably, parameters impacting the characteristics of offline trajectories, as mentioned earlier, remain unchanged. This feature provides operators with the flexibility to actively fine-tune ATL's performance based on the conditions of the scenario.

4.5 Conclusion

Overall, this section has explained in detail all phases of the Adaptive Trajectory Planner (ATL). Addressing the complexities of kinodynamic planning, efficient utilisation of computational resources, and the intricacies of autonomous navigation in unstructured outdoor environments, each phase of ATL has been developed to meet and overcome these challenges. By introducing an offline library storing feasible trajectories, integrated with a comparison process to determine optimal trajectories, the ATL approach appears as a robust and adaptive solution for local planning.

Chapter 5

Experiments

In this chapter, we showcase the practical aspects of our methodology. This validation serves to underscore the robustness and applicability of our approach in diverse settings, proving its effectiveness beyond theoretical constructs. We experimentally demonstrate our approach in both simulation and field trials. The experiment section serves as a critical component in understanding and evaluating the practical implications of the proposed methodology.

5.1 Overview

This section presents an overview of the experimental setup, detailing the key variables and methodologies employed. Our objective is to provide transparency into the testing environment and procedures, enabling readers to assess the validity and reliability of the obtained results.

Commencing with Section 5.2, we explore the physical dimensions characterising the two distinct generations of FORV. Section 5.2.3 delves into the internal system architecture of our platforms, the Fast Off-Road Vehicles (FORV). Within this section, we offer comprehensive insights into the primary components that construct the FORV. The subsequent Section 5.3 describes the construction of offline trajectories utilising the ATL algorithm (1). A special focus is placed on the diverse trajectory libraries tailored for each drive

mechanism integrated into the platform. Following this, Section 5.4 briefly details the simulation testing procedures, the chosen terrain and comparison methodologies employed in the experiments. An explanation of how we set up ATL for the platform during the experimental runs is also presented. Section 5.5 demonstrates the performance of each of the comparison methods, where we showcase the difference between each planner and their effectiveness throughout the experiments. Lastly, Section 5.6 presents the deployment of ATL into the physical FORV1 platform for field experiments. These real-world trials show the capability of ATL in autonomous navigation, highlighting its contribution to an off-road platform.

5.2 Experiment Platform

5.2.1 FORV1

The first generation of *Fast Off-Road Vehicle* (FORV), developed at the University of Technology Sydney (Figure 1.2), was the initial target platform for our simulation. FORV1 measures 2.4 m $\times$ 2 m $\times$ 2 m with a maximum design weight of 400 kg, and drives in a skid-steer configuration where each pair of wheels on each side are given the same command velocity. The design enables high-speed traversal of challenging off-road terrains, while containing desktop-level computing hardware to allow complex control and planning algorithms to run at high frequencies. The software is built using ROS Noetic to create a modular architecture that can be extended for different applications.

5.2.2 FORV2

As we approach the deployment phase of FORV2, our second-generation Fast Off-road Robotic Vehicle as shown in Figure 5.1, we look forward to conducting a performance analysis of ATL. The configuration of FORV2 introduces an advanced four-wheel steering configuration, enabling the robot to traverse in any direction. This significant enhancement over the first version, limited by a constrained turning radius at low speeds, opens up new possibilities for versatile and agile robotic operations.

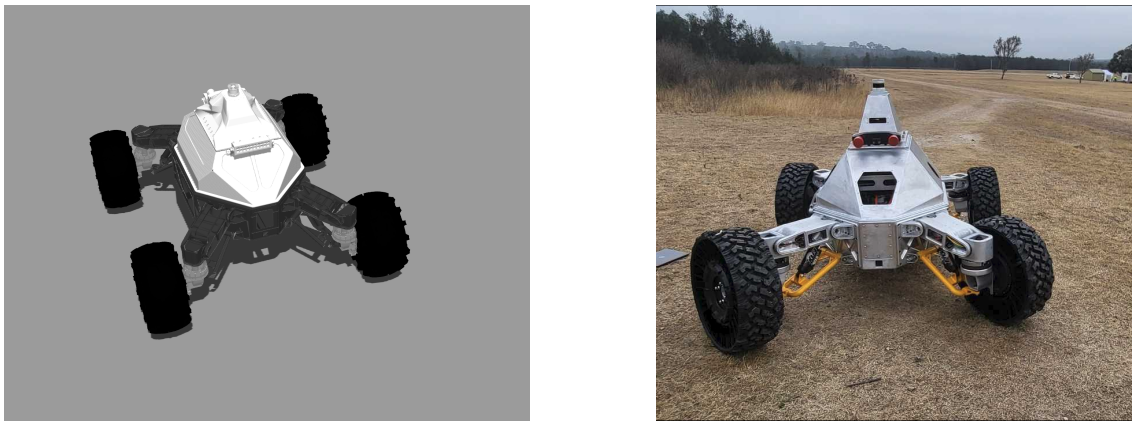


FIGURE 5.1: The FORV 2 developed at UTS in (a) simulation and (b) hardware.

5.2.3 Broader Software System

To test the contributions of this thesis in hardware, many other components were developed to enhance the FORVs platform. Both of the platforms share the same system design, illustrated in Figure 5.2, which shows the architecture.

At the top of the system is the global planner, a crucial element responsible for determining the operation's objectives and waypoints, considering the area's overall map. Occupying the role within the local planner is ATL, tasked with executing real-time behaviors to guide the platform across the area autonomously while prioritising safety and collision avoidance. Lastly, ATL's commands are transmitted to the platform's controller, where they are translated into movements for the robot. The structure of the controller can be divided into three main components: the Joystick Controller, the PLC and the Motor Controllers.

- **Joystick Controller**: The communication setup of FORV involves the communication line from a joystick controller. Two integral components, the controller and the receiver, establish communication through radio signals. Additionally, the controller and receiver interface with the main system through a CAN connection. This configuration facilitates reliable communication between the joystick controller and the receiving unit, ensuring effective integration with the broader system via the CAN connection.

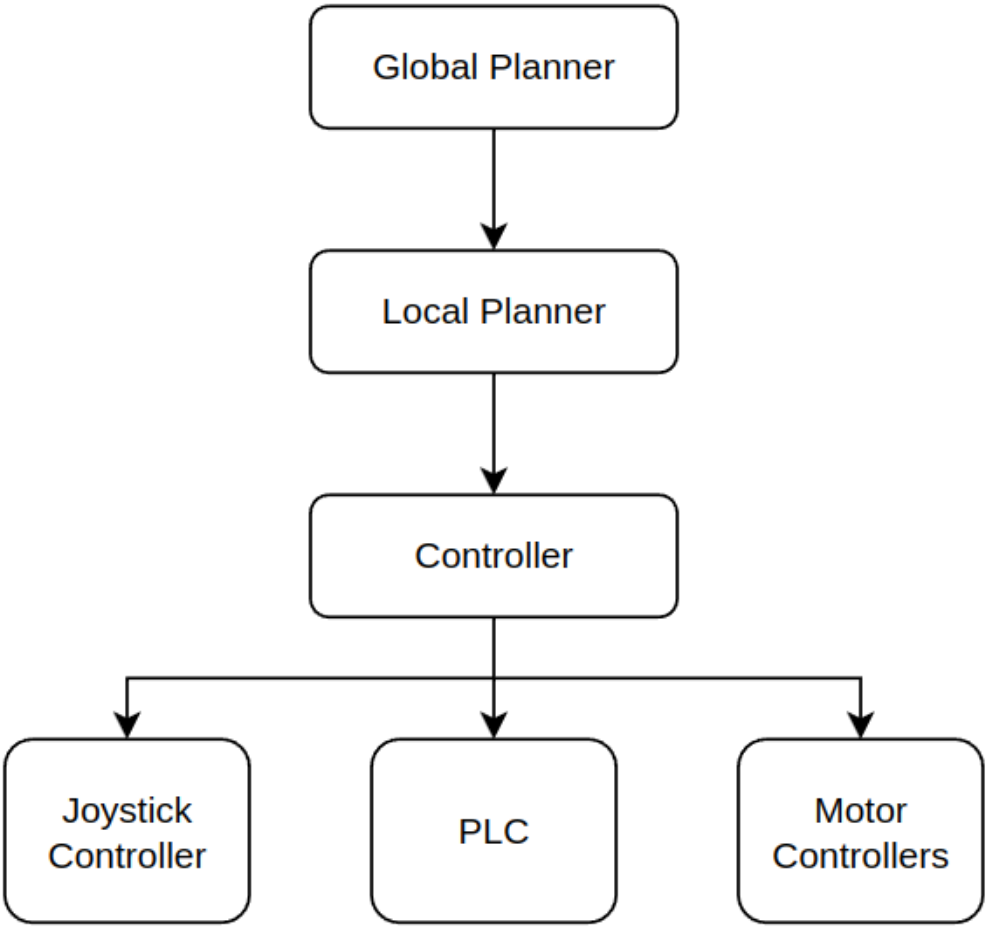


FIGURE 5.2: FORV's system architecture

- **PLC**: The Programmable Logic Controller (PLC) serves as the central hub of the FORV system, managing the electrical processes and regulating signal distribution to each component. Operating at various stages, such as startup, operation, and idle states, the PLC contributes to the FORV's robustness, efficiency, and safety during movements. Functioning as the "heart" of the system, the PLC plays a pivotal role in powering the FORV.

  Attached to the PLC is a Human Machine Interface (HMI) device that provides operators with an intuitive display to monitor the status of FORV components. This interface offers real-time visualisation of crucial information, including battery levels, current status, and motor controller information. Operators can conveniently

observe and configure individual components directly through the display, enhancing control and ensuring efficient operation of the FORV system.

- **Motor Controllers**: FORV is equipped with four brushless DC motors, each under the control of a corresponding Roboteq controller. These controllers establish communication with a Neusys POC system, an industrial-grade mini-PC specifically employed to manage these motor controllers. Utilising the CAN protocol, the motor controllers interface with the main system, ensuring the robust and reliable exchange of signals and information. This standard communication protocol enhances the dependability of the overall system, allowing for efficient coordination and control of the FORV's motorised components.

FORV is equipped with the most advanced components, ensuring every operation's reliability and quality for this fast off-road ground robot. To fully unlock the potential of this platform in autonomous navigation, a robust and effective local planner is essential. This planner must be suitable and highly efficient, leveraging the cutting-edge features of FORV to ensure precision and reliability in every operation. Thus, we investigate the impact of ATL on the performance of FORV for autonomous navigation tasks.

5.3 ATL Trajectory Library

As mentioned in Chapter 4, ATL is designed to integrate a dedicated dynamically feasible trajectories library for a target platform. Given the distinct driving configurations of the two FORV versions—differential drive and four-wheel steering drive—ATL is assigned the responsibility of generating a trajectories library that aligns with the unique dynamics requirements of each platform.

5.3.1 FORV1

Following the characteristics of FORV1, the drive mechanism performs movements based on two separately driven wheels placed on either side of the robot body. This design allows the robot to alter its direction by adjusting the relative rotation speed of its wheels,

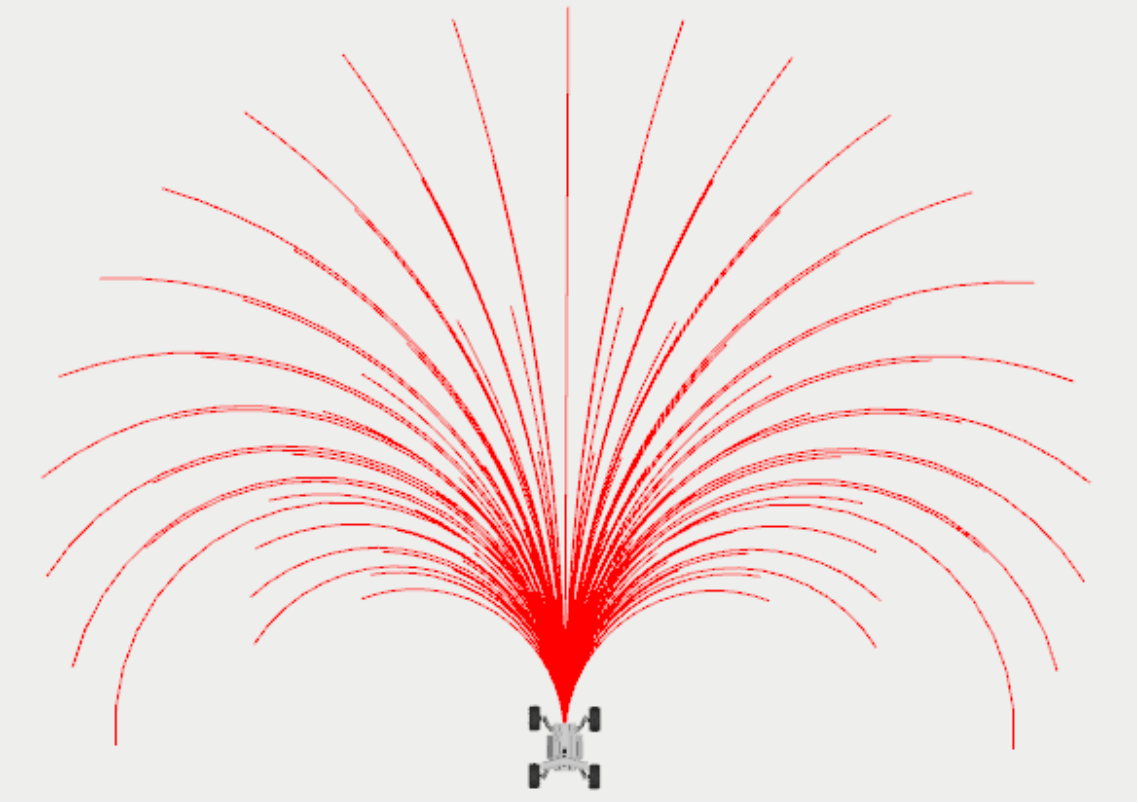


FIGURE 5.3: FORV1 Offline Trajectories Library

eliminating the need for an extra steering mechanism. ATL implements a differential drive kinematic equation to generate offline trajectories as mentioned in Section 4.2. The library consists of collections with forward velocity $v \in \{1, 1.5, 2, 3, 4\}$ m/s and angular velocity ω °/s, where:

$$\omega \in \begin{cases} [-10, 10] \text{ step } 2, & v = 1 \\ [-12, 12] \text{ step } 3, & v = 1.5 \\ [-20, 20] \text{ step } 4, & v = 2 \\ [-28, 28] \text{ step } 4, & v = 3 \\ [-36, 36] \text{ step } 9, & v = 4 \\ [-50, 50] \text{ step } 10, & v = 5 \end{cases}$$

The trajectories library of FORV1 can be visualised via the following Figure 5.3.

The steps associated with the angular velocities denote the spacing or increment between

consecutive values within each specified range. For example, [-10, 10] step 2 represents a range from -10 to 10, where each step between consecutive values is 2 units [-10, -8, -6, -4, -2, 0, 2, 4, 6, 8, 10].

These trajectories highlight the capability of generating precise estimation trajectories for ATL's platform. This library is utilised for both simulation experiments and hardware experiments.

5.3.2 FORV2

The drive mechanism of FORV2 employs a four-wheel steering system, providing advantages such as a short turning radius and preventing wheel slippage during rotation. Consequently, it becomes crucial to develop a dedicated library capable of adapting to the specific capabilities of FORV2, thereby exploiting the full potential of the platform's design. By adding information for horizontal movements, ATL enhances the library's performance with the benefit of diagonal trajectories. These trajectories are specifically designed to accommodate the capabilities of FORV2, allowing the platform to execute them effortlessly. To exploit the ability of the four-wheel steering configuration from FORV2, we implemented a set of horizontal linear velocities to enable the platform with the ability to execute "crabbing" motion, where the robot can travel horizontally. Similar to the FORV1 set-up, the configuration file of FORV2 can be listed as:

$$\omega \in \begin{cases} [-10, 10] \text{ step } 2, & v_x = 1, v_y = [-1, 1] \text{ step } 0.5 \\ [-12, 12] \text{ step } 3, & v_x = 1.5, v_y = [-2, 2] \text{ step } 0.2 \\ [-20, 20] \text{ step } 4, & v_x = 2, v_y = 0 \\ [-28, 28] \text{ step } 4, & v_x = 3, v_y = 0 \\ [-36, 36] \text{ step } 9, & v_x = 4, v_y = 0 \\ [-50, 50] \text{ step } 10, & v_x = 5, v_y = 0 \end{cases}$$

The trajectories library of FORV2 can be visualised via the following Figure 5.4.

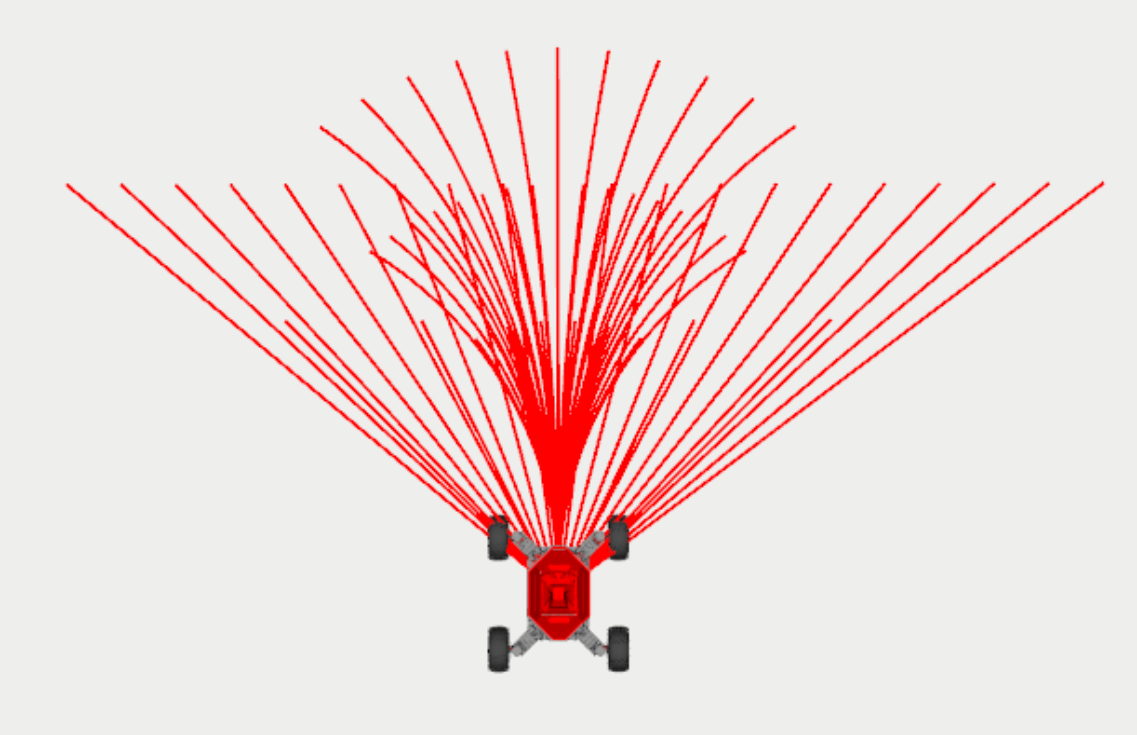


FIGURE 5.4: FORV2 Offline Trajectories Library

This demonstrates ATL's versatility as a solution compatible with diverse drive mechanisms. By customised ATL parameters, operators can easily incorporate feasible trajectories tailored to the platform's capabilities.

5.4 Simulation Setup

5.4.1 Scenario

The ATL planner was first evaluated in simulation to determine its suitability for high speed operation in outdoor environments. Simulations were run in Gazebo, a popular simulation package in the robotics community that enables accurate modelling of complex real-world scenarios. The simulated FORV was loaded into a terrain based off that provided in the NEGS-UGV Dataset (Sánchez et al., 2022). The environment measures 100 m $\times$ 50 m, and consists of a flat surface with obstacles such as trees, benches, and street lamps distributed throughout (Figure 5.5). The spacing between obstacles was chosen to provide both open spaces and areas where the robot would be required to negotiate tight

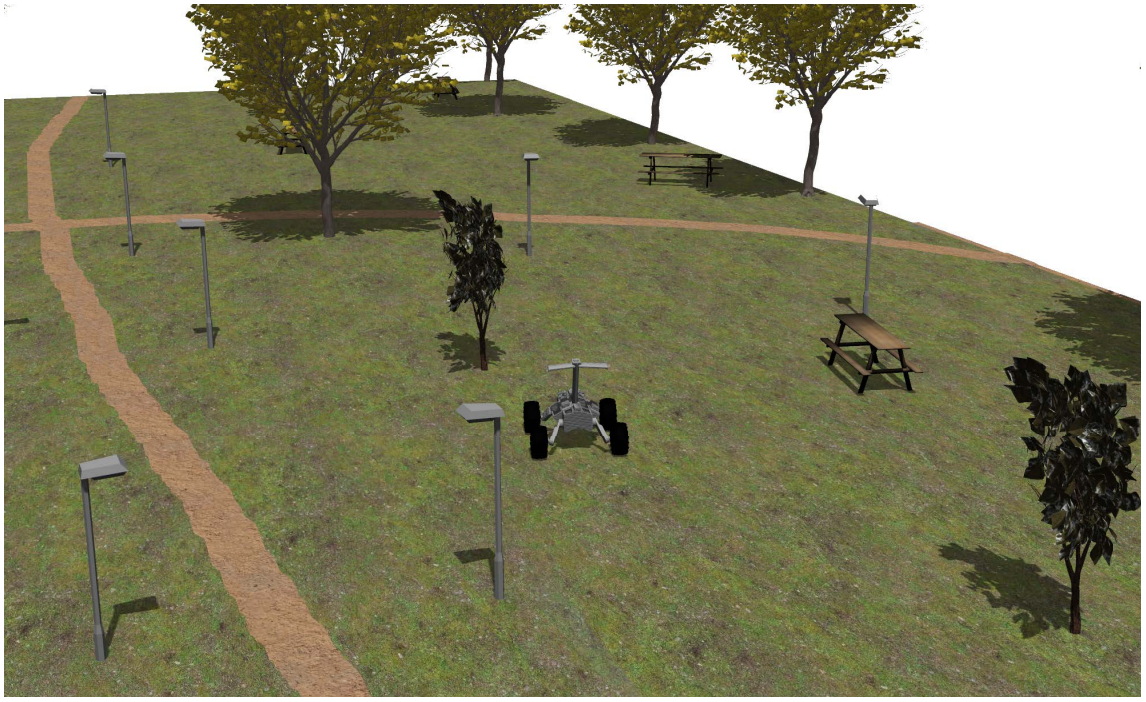


FIGURE 5.5: FORV within the NEGS-UGV park terrain used in Gazebo simulations.

gaps. In order to examine the performance of the planners independent of other factors, the robot obtains its location directly from Gazebo instead of relying on internal odometry, and a static costmap was generated from the terrain offline using the `costmap_2d` package (Marder-Eppstein et al., 2023).

FORV was initialised near the top right corner of the simulated environment, and provided with several ordered waypoints to visit; waypoints were marked as visited when the robot reached a location within 3 m of them. The simulation setup for FORV2 and ATL mirrors the experimental conditions employed in the first generation. We used the identical simulation terrain featured in the NEGS-UGV Dataset, along with the consistent deployment of the same waypoints.

5.4.2 Comparision Methods

Our comparison methods are:

- **ATL Planner**: ATL was implemented with two sets of dynamic thresholds as mentioned in Section 4.3.1 at $\mathbf{k}^x = 3$ m/s and $\mathbf{k}^w = 2$ rad/s and $\mathbf{k}^x = 2$ m/s and $\mathbf{k}^w =$

1 rad/s for linear and angular velocities. The ATL planner is designed to determine the optimal control action from the provided dynamically feasible trajectories based on these dynamic thresholds.

- **DWA Planner**: Dynamic Window Approach (DWA) planner was implemented using the `dwa_local_planner` package from ROS (Christian Connette, 2023a). DWA is known for its capability to generate feasible trajectories using a dynamically feasible window to produce velocity commands to send to the platform.

- **TEB Planner**: Timed Elastic Band (TEB) planner was implemented using the `teb_local_planner` package from ROS (Rösmann, 2023). TEB planner is suitable for systems with non-holonomic constraints and takes into account the robot's kinematics and dynamic constraints while planning trajectories.

- **A* Planner**: A* algorithm is implemented via the `global_planner` package (Christian Connette, 2023b). The planner utilises the A* algorithm for global path planning, and it was complemented by a Pure Pursuit (Coulter, 1992a) controller. The Pure Pursuit controller adjusts the robot's velocity based on the waypoints generated by the A* planner.

Fifty simulations were performed for each planner, and the paths followed by FORV were recorded for later analysis.

5.4.3 Implementation Details

The ATL planner is implemented with the trajectory library config in a `YAML` file loaded at runtime, as mentioned in Section 4.4. These values were chosen through empirical observations of FORV driving around a testing environment. The allowable ω is increased at greater v as we have observed that the skid steer dynamics of FORV allow it to perform tighter turns when travelling at higher speeds. Each trajectory is simulated for 5 s with points sampled at intervals of 0.2 s along it to determine whether it results in a collision and to evaluate the cost function.

The DWA Planner parameters are configured with the following values:

Parameter	**Values**	**Units**
Holonomic Robot	False	-
Sim Time	10.0	s
Sim Granularity	0.2	s
Angular Sim Granularity	0.1	rad
Vx Samples	20	-
Vtheta Samples	30	-
Controller Frequency	10.0	Hz
Meter Scoring	True	-
Occdist Scale	0.5	-
Pdist Scale	2	-
Gdist Scale	5	-
Heading Lookahead	2.0	m
Heading Scoring	True	-
Heading Scoring Timestep	1.0	s
DWA	True	-

TABLE 5.1: DWA Planner Parameters

The TEB Planner parameters are configured with the following values:

The A* parameters are configured with the following values:

For A* Planner, the attached Pure Pursuit controller has the same limitation values for acceleration and velocities.

5.5 Simulation Results

5.5.1 FORV1

The paths followed by FORV in each trial are plotted in Figure 5.6, and summarised in Table 5.4. It can be seen that our ATL planner, at two different dynamic threshold sets (3 & 2 and 2 & 1 for linear and angular velocities), performs favourably compared to the DWA, TEB and A* planners. In summary, among the considered techniques, DWA is the least efficient for the given operation, showcasing the longest duration, travel distance, and average planning time per iteration. Notably, paths generated by the alternative methods are considerably shorter than those of DWA, with comparable lengths. These paths can be travelled faster than those the DWA planner produces. Graph search technique A*

Parameter	Values	Units
No Inner Iterations	5	-
No Outer Iterations	4	-
Optimization Activate	True	-
Optimization Verbose	False	-
Penalty Epsilon	3.0	-
Weight Max Vel X	1	-
Weight Max Vel Theta	1	-
Weight Acc Lim X	0.1	-
Weight Acc Lim Theta	0.1	-
Weight Kinematics NH	1000	-
Weight Kinematics Forward Drive	1000	-
Weight Kinematics Turning Radius	1	-
Weight Optimal Time	1.0	-
Weight Obstacle	50	-
Weight Viapoint	0.4	-
Weight Inflation	1.0	-
Weight Dynamic Obstacle	10	-
Selection Alternative Time Cost	False	-

Table 5.2: TEB Planner Parameters

Parameter	Values	Units
Allow Unknown	True	-
Default Tolerance	0.0	-
Visualize Potential	True	-
Use Dijkstra	False	-
Use Quadratic	True	-
Use Grid Path	False	-
Old Navfn Behavior	False	-
Lethal Cost	253	-
Neutral Cost	0	-
Cost Factor	3	-
Publish Potential	True	-
Orientation Mode	0	-
Orientation Window Size	1	-
Outline Map	True	-

Table 5.3: A* Parameters

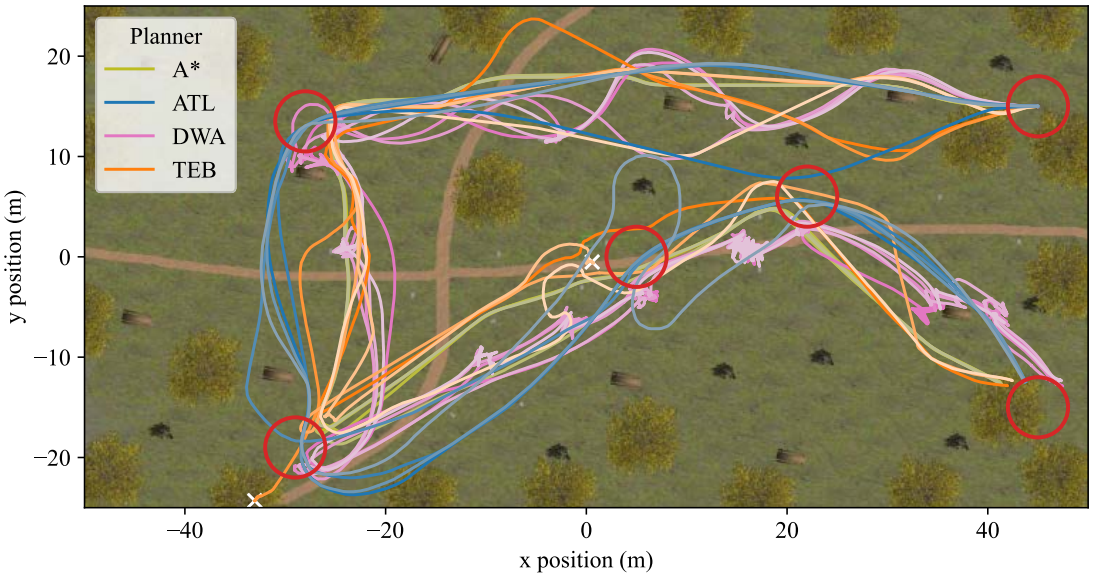


FIGURE 5.6: Simulation results for the ATL, DWA, TEB and A* planners. In each trial, the robot starts on the top right and moves through the waypoints in red, where the circle indicates the acceptance radius. Trajectories followed by FORV under each planner are shown in blue, orange, and green with different trials indicated by different saturations.

Planner	**Travel time (s)**	**Path length (m)**	**Planning time (ms)**
ATL$(2, 1)$	97	228	21
ATL$(3, 2)$	90	205	26
DWA	569	339	198
TEB	170	223	19
A*	188	183	30

TABLE 5.4: Simulation results for the ATL, DWA, TEB and A* planners. Values given are the mean values across multiple trials.

produces paths with the shortest travel distance and time. However, their average computational time remains higher than TEB and ATL, as the characteristics of these methods demand a certain amount of time to determine the optimal solution.

ATL and TEB demonstrate their effectiveness by generating smooth paths featuring wide corners, facilitating high-speed traversal. In contrast, DWA frequently opts for tight or on-the-spot turns, reducing the linear speed achievable by the robot. It is worth mentioning that the real FORV has limitations in executing such tight cornering turning, thereby restricting the practicality of DWA on our selected platform.

The success rate achieved by each planner is another important consideration. Trials can fail when the planner is unable to generate control commands, the robot collides with an obstacle, or the robot becomes immobilised due to its footprint overlapping with an obstacle zone within the costmap. All trials of the ATL, DWA and A* planners succeeded, but approximately half of the TEB planners failed. In proximity to densely populated obstacle areas, TEB generated inefficient commands, often outputting large angular velocities. This tendency caused the robot to oscillate around obstacles in an attempt to find a viable path through the area, eventually driving it into occupied regions of the costmap from which it could not recover.

On the other hand, by adjusting the dynamic threshold, we can see a slight improvement in the performance of the ATL, highlighting the need for the filtering process mentioned in Section 4.3.1. A small dynamic threshold can reduce the computational cost for the system, yet this affects the performance of ATL for the task. As we can see the travel time and path length have increased as the small threshold filter more infeasible trajectories, making the size of the dynamic library smaller. Consequently, this limits the options available during the matching phase, making it more challenging to identify the optimal solution.

The planners perform relatively consistently during successful trials, moving through similar trajectories. Occasionally, obstacles would be passed on the opposite side to what was usually selected by that planner, but the paths would soon converge again. This can lead to a ripple effect, where passing one obstacle on a different side results in a longer path overall as further obstacles must be avoided while obeying dynamic feasibility constraints before returning to the usual path.

In addition to the metrics relating to the paths produced, the computational cost of each planner was also compared. We measured the time taken to calculate the velocity commands at each planning instant, and found the average planning time of our ATL planner was significantly lower than DWA but slightly higher than TEB. This shows the low computational cost of our method in producing successful paths, which lends itself to the fast planning required for robots travelling at high speeds.

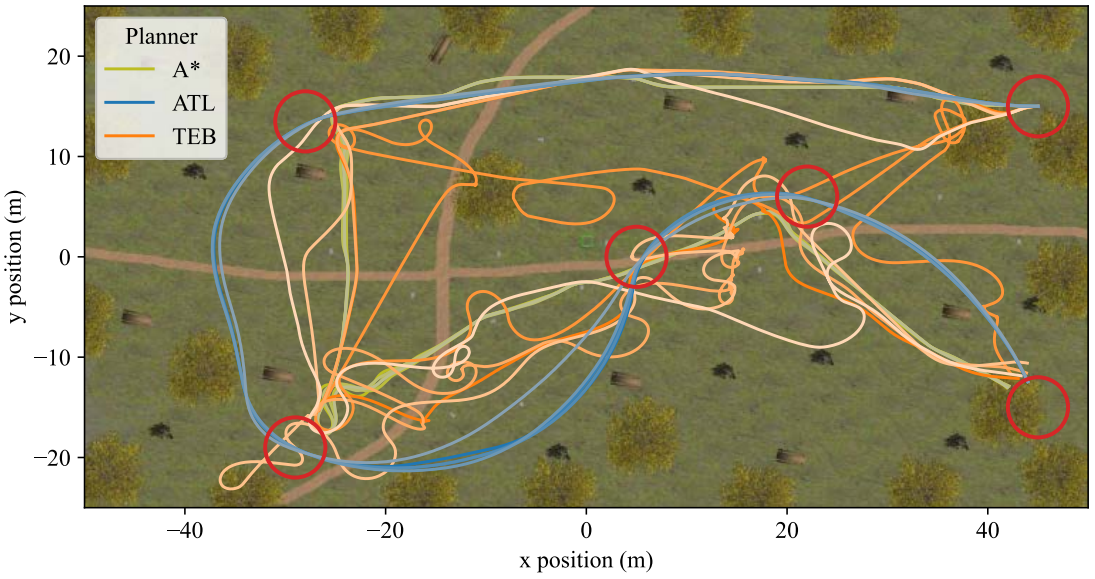


FIGURE 5.7: Simulation results for the ATL, TEB and A* planners. In each trial, the robot starts on the top right and moves through the waypoints in red, where the circle indicates the acceptance radius. Trajectories followed by FORV2 under each planner are shown in blue, orange, and green with different trials indicated by different saturations.

5.5.2 FORV2

For the FORV2 Simulation, we selected four planners: ATL, TEB and A* to conduct the comparison experiment. Similar to the first experiment, the paths followed by FORV 2 in each trial are plotted in Figure 5.7, and summarised in Table 5.5

The four-wheel steering configuration of FORV2 presents a distinct advantage, allowing the platform to achieve a higher turning radius and improve its maneuverability easily. Recorded data from Table 5.5 has shown the improved performance of the ATL planner when subjected to dynamic threshold sets (2 & 1 for linear and angular velocities).

FORV2, being controlled by ATL, travelled a shorter distance compared to FORV1, achieving an approximate 10% reduction in both travel distance and time. An increment in planning time can be seen as the result of the additional horizontal crabbing trajectories, which the platform's configuration can execute. The ATL planner showcases the ability to adapt to changing conditions, resulting in more efficient navigation.

Planner	Travel time (s)	Path length (m)	Planning time (ms)
ATL(2, 1)	88	206	24
TEB	168	211	18
A*	186	188	31

TABLE 5.5: Simulation results for the ATL, A*, and TEB planners. Values given are the mean values across multiple trials.

On the other hand, the TEB planner presented a step down in performance compared to its earlier result. The challenges are evident as FORV2 struggled to align to the commanded path, often looping around positions close to environmental obstacles. This disadvantage underscores the planner's limitations in dynamically adapting to the platform's configuration, raising questions regarding its suitability for certain scenarios.

Meanwhile, the A* method produces consistent results compared to their performance with the previous FORV1 experiments. Although the observed increment in the travelled path is trivial, it is considerable that the planner maintains a level of reliability.

In conclusion, this comparison reveals the strengths and weaknesses of each planner in the compatibility with FORV2's capabilities. The ATL planner stands as a standout performer, balancing between computational time usage and travelled distance/time, leveraging the advantages of four-wheel steering to achieve better performance. In contrast to ATL, the TEB planner struggles to maintain robust performance, highlighting the importance of selecting planners based on specific operational requirements. The performance observed in the A* method confirms their reliability, bringing attention to their adaptability across multiple simulation scenarios.

5.6 Field Experiments

Our ATL planner was also tested on the physical FORV1 platform to evaluate its real-world performance. Broadly the same software was run on the real robot as in simulation, however it was not possible to isolate the performance of the planner from the localisation and costmap generation. In real-world experiments, the robot calculates its position by fusing data from motor encoders, an inertial measurement unit, and GPS using

the `robot_localization` package for ROS (Moore and Stouch, 2014). Similarly a static costmap cannot be used, and instead a live costmap is generated from lidar data.

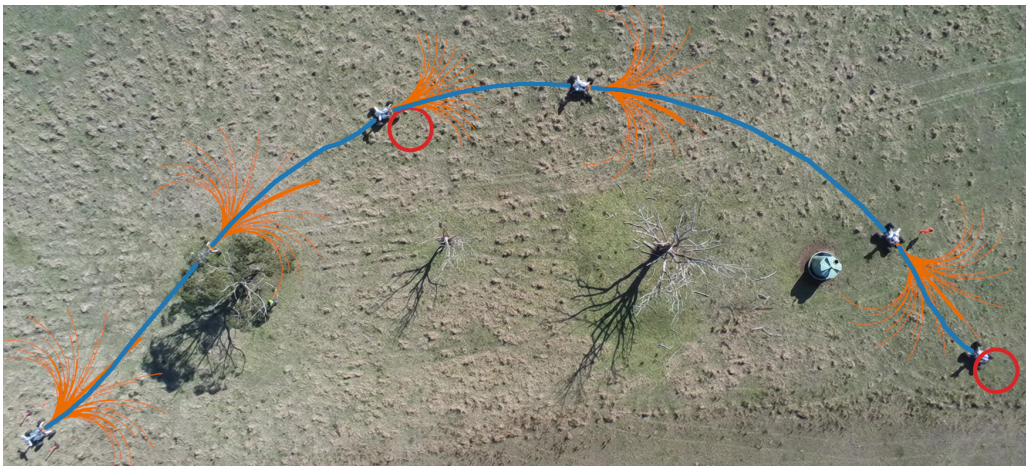
5.6.1 Experiments Setup

We conducted a field trial at Sydney Science Park during which two experiments were performed. In each experiment, FORV1 began near a selection of obstacles including trees and a dummy human. FORV1 was instructed to move through two waypoints with a tolerance of 3 m then stop. Two experiments were performed, varying the starting position and orientation of FORV1 with respect to the obstacles as well as the position of the waypoints to determine the performance of the planner in different scenarios. We set the same dynamic threshold $\mathbf{k}^x = 3\text{m/s}$ and $\mathbf{k}^w = 115$ degrees for the filtering process as the simulation experiments.

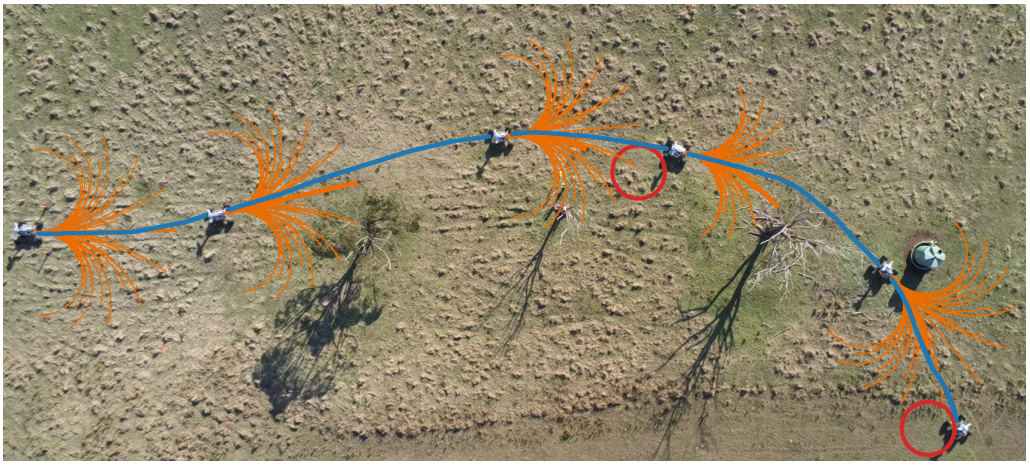
5.6.2 Result

Results from the field trial are shown in Figure 5.8. In the first experiment (Figure 5.8a), FORV1 was positioned a short distance from a tree and oriented directly towards it. A waypoint was placed directly behind this tree, such that FORV1 would have to avoid an obstacle immediately. Another waypoint was placed a short distance away to direct FORV1 to turn and avoid two further obstacles. The ATL planner performed well in this scenario, avoiding all obstacles and moving through a smooth path without sharp turns that would reduce its linear velocity. The robot travelled 96 m in 33 s, achieving an average linear speed of 2.9 m/s.

The second experiment considered a more challenging scenario (Figure 5.8b). FORV1 was initially aligned pointing directly down a row of obstacles, and waypoints were selected to direct the robot to slalom between them. This was achieved without any collisions, demonstrating the impressive obstacle avoidance capabilities of the planner even when operating at relatively high speeds. In this trial, the robot travelled 136 m in 28 s, so moved with a higher average linear speed than the previous trial of 4.9 m/s.



(A)



(B)

FIGURE 5.8: Real-world experimental results. Waypoints are shown in red, where the circle indicates the acceptance radius. The trajectories followed by FORV are plotted in blue. FORV is shown at several points equally spaced in time (5.5 s in (a) and 6.6 s in (b)). At each of these points, the current feasible trajectories (orange) and the selected path (thick orange) are also visualised.

5.7 Conclusion

In conclusion, this section has highlighted ATL's performance through its results in both simulation and hardware experiments, offering valuable insights into its capabilities under diverse conditions of the experiment scenario. The adaptability and effectiveness of ATL in the environment set up with two different platforms have showcased its potential as a robust and versatile solution for addressing complex scenarios.

Chapter 6

Conclusion

This thesis has presented ATL, a novel local path planning algorithm suitable for robots travelling at high speeds. The approach reduces online computation by using a precomputed trajectory library, an adaptive subset of which is sampled to reflect trajectories that are dynamically feasible given the current state of the robot.

6.1 Summary of Contribution

To sum up, addressing the complexities of autonomous navigation for a high-speed off-road mobile robot requires innovative solutions to overcome the challenges. Traditional navigation methods often face limitations in handling the rapid and unpredictable changes of the platform's movement, underscoring the need for advanced methodologies. This thesis presents the Adaptive Trajectory Library (ATL) as a local planning solution, addressing the three key major points highlighted in the introduction: performance in unstructured environments, efficiency in computational usage, and kinodynamic planning capabilities.

Chapter 4 of this thesis describes ATL's core algorithm and operational sequence, presenting how it effectively addresses the complexities associated with high-speed off-road navigation. ATL represents a shift in autonomous navigation strategies by integrating a trajectories library from a set of feasible velocities gathered from a target platform. This results in providing the planner with the ability to control the robot with dynamically

feasible velocities, enhancing the performance of navigating dynamic terrains at elevated speeds. Furthermore, ATL's core is designed to adapt to real-time platform feedback, allowing it to control the platform dynamically. By doing so, ATL can make instantaneous decisions, choosing an efficient and feasible path to follow the commanded path throughout the landscape.

In conclusion, introducing the ATL in this thesis marks a significant step forward in addressing the intricate challenges of autonomous navigation in high-speed off-road scenarios. ATL represents a novel approach that holds promise for enhancing the capabilities of existing robots and influencing the design and development of future generations of autonomous mobile platforms. As the field continues to evolve, the insights and methodologies presented in this thesis contribute valuable knowledge to the broader discourse on advancing autonomous robotics.

6.2 Future Work

Our future research will focus on investigating the improvements and expansion of the trajectory library to accommodate wider fields of real-world constraints. Recognising that the robot's dynamics are linked to the terrain characteristics it traverses, we aim to enhance the trajectory library to enable dynamic variations in trajectories based on the diverse surfaces encountered. This adaptation will allow the robot to navigate with greater adaptability, optimising its movements according to the specific challenges posed by different terrains.

Furthermore, an aspect of our future investigations involves the integration of ATL with a global planner. This strategy aims to harness the strengths of both systems, creating a comprehensive approach to autonomous navigation. The global planner will be designed to generate waypoints strategically, providing a road map for the robot to travel to larger areas autonomously.

By exploring these features, we aspire to push the boundaries of ATL and autonomous robotics, creating a planner that adapts to the dynamic nature of its surroundings and possesses the intelligence to navigate across extensive terrains. This research contributes

to the advancement of autonomous mobile robots and opens up new possibilities for applications in fields ranging from environmental monitoring to infrastructure inspection.

Bibliography

Ammar, A., Bennaceur, H., Châari, I., Koubâa, A., and Alajlan, M. (2016). Relaxed dijkstra and a* with linear complexity for robot path planning problems in large-scale grid environments. *Soft Comput.*, 20(10):4149–4171.

Bardi, M. and López, J. P. M. (2015). A dijkstra-type algorithm for dynamic games. *Dynamic Games and Applications*, 6:263 – 276.

Becker, J., Imholz, N., Schwarzenbach, L., Ghignone, E., Baumann, N., and Magno, M. (2023). Model- and acceleration-based pursuit controller for high-performance autonomous racing. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE.

Best, G., Garg, R., Keller, J., Hollinger, G. A., and Scherer, S. (2023). Multi-robot, multi-sensor exploration of multifarious environments with full mission aerial autonomy. *International Journal of Robotics Research*.

Bjelonic, M., Grandia, R., Geilinger, M., Harley, O., Medeiros, V. S., Pajovic, V., Jelavic, E., Coros, S., and Hutter, M. (2022). Offline motion libraries and online mpc for advanced mobility skills. *The International Journal of Robotics Research*, 41(9-10):903–924.

Bonalli, R., Cauligi, A., Bylard, A., and Pavone, M. (2019). Gusto: Guaranteed sequential trajectory optimization via sequential convex programming.

Campbell, S. F. (2007). *Steering control of an autonomous ground vehicle with application to the DARPA urban challenge*. PhD thesis, Massachusetts Institute of Technology.

Carroll, M. (2023). dynamic_reconfigure. `https://wiki.ros.org/dynamic_reconfigure`.

Cetin, O., Ozmen, B., Kurnaz, S., and Temeltas, H. (2009). Potential field based navigation task for autonomous flight control of unmanned aerial vehicles. volume 5.

Chen, J.-T., Yousefi, A., Krishna, S., Sliney, B., and Smith, P. (2012). Weather avoidance optimal routing for extended terminal airspace in support of dynamic airspace configuration. In *2012 IEEE/AIAA 31st Digital Avionics Systems Conference (DASC)*, pages 3A1–1–3A1–16.

Choi, B., Kim, B., Kim, E., and Yang, K. W. (2012). A modified dynamic window approach in crowded indoor environment for intelligent transport robot. In *2012 12th International Conference on Control, Automation and Systems*, pages 1007–1009.

Christian Connette, Bhaskara Marthi, P. K. (2023a). dwa_local_planner. `http://wiki.ros.org/dwa_local_planner`.

Christian Connette, Bhaskara Marthi, P. K. (2023b). global_planner. `http://wiki.ros.org/global_planner`.

Chu, K., Lee, M., and Sunwoo, M. (2012). Local path planning for off-road autonomous driving with avoidance of static obstacles. *IEEE Transactions on Intelligent Transportation Systems*, 13(4):1599–1616.

Chung, Y. M., Youssef, H., and Roidl, M. (2022). Distributed timed elastic band (dteb) planner: Trajectory sharing and collision prediction for multi-robot systems. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 10702–10708.

Coulter, C. (1992a). Implementation of the pure pursuit path tracking algorithm.

Coulter, R. C. (1992b). *Implementation of the pure pursuit path tracking algorithm*. Carnegie Mellon University, The Robotics Institute.

Cybulski, B., Wegierska, A., and Granosik, G. (2019). Accuracy comparison of navigation local planners on ros-based mobile robot. In *2019 12th International Workshop on Robot Motion and Control (RoMoCo)*, pages 104–111.

Dijkstra, E. W. (1959). A note on two problems in connexion with graphs. *Numerische Mathematik*, 1:269–271.

Donald, B., Xavier, P., Canny, J., and Reif, J. (1993). Kinodynamic motion planning. *J. ACM*, 40(5):1048–1066.

Fan, D. and Shi, P. (2010). Improvement of dijkstra's algorithm and its application in route planning. In *2010 Seventh International Conference on Fuzzy Systems and Knowledge Discovery*, volume 4, pages 1901–1904.

Ferguson, D. and Stentz, A. (2007). Field d*: An interpolation-based path planner and replanner.

Fox, D., Burgard, W., and Thrun, S. (1997). The dynamic window approach to collision avoidance. *IEEE Robotics & Automation Magazine*, 4(1):23–33.

Frazzoli, E., Dahleh, M., and Feron, E. (2001). Real-time motion planning for agile autonomous vehicles. In *Proceedings of the 2001 American Control Conference. (Cat. No.01CH37148)*, volume 1, pages 43–49 vol.1.

Goswami, A. (2017). *Hierarchical Off-Road Path Planning and Its Validation Using a Scaled Autonomous Car*. PhD thesis, Clemson University.

Grothe, F., Hartmann, V. N., Orthey, A., and Toussaint, M. (2022). St-rrt*: Asymptotically-optimal bidirectional motion planning through space-time. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 3314–3320.

Hart, P. E., Nilsson, N. J., and Raphael, B. (1968). A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions on Systems Science and Cybernetics*, 4(2):100–107.

Hsu, D., Sanchez-Ante, G., and Sun, Z. (2005). Hybrid prm sampling with a cost-sensitive adaptive strategy. In *Proceedings of the 2005 IEEE International Conference on Robotics and Automation*, pages 3874–3880.

Huang, L. and Yi, H. (2021). Research on the optimal strategy of "crossing the desert" game based on dijkstra algorithm. *Academic Journal of Computing & Information Science*.

Hönig, W., Ortiz-Haro, J., and Toussaint, M. (2022). db-a*: Discontinuity-bounded search for kinodynamic mobile robot motion planning. In *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 13540–13547.

Jaillet, L. and Simeon, T. (2004). A prm-based motion planner for dynamically changing environments. In *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (IEEE Cat. No.04CH37566)*, volume 2, pages 1606–1611 vol.2.

Jian, X., Zou, T., Vardy, A., and Bose, N. (2020). A hybrid path planning strategy of autonomous underwater vehicles. In *2020 IEEE/OES Autonomous Underwater Vehicles Symposium (AUV)*, pages 1–6.

Joglekar, A., Sathe, S., Misurati, N., Srinivasan, S., Schmid, M. J., and Krovi, V. (2022). Deep reinforcement learning based adaptation of pure-pursuit path-tracking control for skid-steered vehicles. *IFAC-PapersOnLine*, 55(37):400–407. 2nd Modeling, Estimation and Control Conference MECC 2022.

Kalakrishnan, M., Chitta, S., Theodorou, E., Pastor, P., and Schaal, S. (2011). Stomp: Stochastic trajectory optimization for motion planning. In *2011 IEEE International Conference on Robotics and Automation*, pages 4569–4574.

Karaman, S. and Frazzoli, E. (2010). Incremental sampling-based algorithms for optimal motion planning.

Karaman, S. and Frazzoli, E. (2011). Sampling-based algorithms for optimal motion planning. *The International Journal of Robotics Research*, 30(7):846–894.

Karur, K., Sharma, N., Dharmatti, C., and Siegel, J. E. (2021). A survey of path planning algorithms for mobile robots. *Vehicles*, 3(3):448–468.

Kavraki, L., Svestka, P., Latombe, J.-C., and Overmars, M. (1996). Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE Transactions on Robotics and Automation*, 12(4):566–580.

Khatib, O. (1986). Real-time obstacle avoidance for manipulators and mobile robots. *International Journal of Robotics Research*, 5(1):90–98.

Khuswendi, T., Hindersah, H., and Adiprawita, W. (2011). Uav path planning using potential field and modified receding horizon a* 3d algorithm. In *Proceedings of the 2011 International Conference on Electrical Engineering and Informatics*, pages 1–6.

Koenig, S. and Likhachev, M. (2001). Incremental a*. In Dietterich, T., Becker, S., and Ghahramani, Z., editors, *Advances in Neural Information Processing Systems*, volume 14. MIT Press.

Koren, Y. and Borenstein, J. (1991). Potential field methods and their inherent limitations for mobile robot navigation. 2:1398–1404.

Kuffner, J. and LaValle, S. (2000). Rrt-connect: An efficient approach to single-query path planning. In *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No.00CH37065)*, volume 2, pages 995–1001 vol.2.

Kumar, S. and Chakravorty, S. (2012). Multi-agent generalized probabilistic roadmaps: Magprm. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3747–3753.

Kuwata, Y., Teo, J., Fiore, G., Karaman, S., Frazzoli, E., and How, J. P. (2009). Real-time motion planning with applications to autonomous urban driving. *IEEE Transactions on Control Systems Technology*, 17(5):1105–1118.

LaValle, S. (1998). Rapidly-exploring random trees: A new tool for path planning. *Research Report 9811*.

LaValle, S. M. and Kuffner Jr, J. J. (2001). Randomized kinodynamic planning. *International Journal of Robotics Research*, 20(5):378–400.

Le, D. and Plaku, E. (2014). Guiding sampling-based tree search for motion planning with dynamics via probabilistic roadmap abstractions. In *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 212–217.

Lee, J. J. H., Frey, K., Fitch, R., and Sukkarieh, S. (2014). Fast path planning for precision weeding. In *Proc. of Australasian Conference on Robotics and Automation (ACRA)*.

Li, J. and Zhang, Z. (2023). Auv local path planning based on fusion of improved dwa and rrt algorithms. In *2023 IEEE International Conference on Mechatronics and Automation (ICMA)*, pages 935–941.

Likhachev, M. and Ferguson, D. (2009). Planning long dynamically feasible maneuvers for autonomous vehicles. *International Journal of Robotics Research*, 28(8):933–945.

Likhachev, M., Ferguson, D., Gordon, G., Stentz, A., and Thrun, S. (2005). Anytime dynamic a*: An anytime, replanning algorithm. In *Proceedings of the Fifteenth International Conference on International Conference on Automated Planning and Scheduling*, ICAPS'05, page 262–271. AAAI Press.

Lu, N., Deng, Q., Zhang, J., and Wang, X. (2022). Ship real-time route planning based on field theory and dynamic window approach. In *2022 5th International Conference on Intelligent Autonomous Systems (ICoIAS)*, pages 109–113.

Lucas, G. (2001). A tutorial and elementary trajectory model for the differential steering system of robot wheel actuators. `https://rossum.sourceforge.net/papers/DiffSteer/`.

Macenski, S., Singh, S., Martín, F., and Ginés, J. (2023). Regulated pure pursuit for robot path tracking. *Autonomous Robots*, pages 685–694.

Magyar, B., Tsiogkas, N., Deray, J., Pfeiffer, S., and Lane, D. (2019). Timed-elastic bands for manipulation motion planning. *IEEE Robotics and Automation Letters*, PP:1–1.

Marder-Eppstein, E., Lu, D. V., and Hershberger, D. (2023). costmap_2d. `http://wiki.ros.org/costmap_2d`.

Melchior, N. A. and Simmons, R. (2007). Particle rrt for path planning with uncertainty. In *Proceedings 2007 IEEE International Conference on Robotics and Automation*, pages 1617–1624.

Melchiorre, M., Salamina, L., Scimmi, L. S., Mauro, S., and Pastorelli, S. (2023). Experiments on the artificial potential field with local attractors for mobile robot navigation. *Robotics*, 12.

Moore, T. and Stouch, D. (2014). A generalized extended Kalman filter implementation for the robot operating system. In *Proc. of International Conference on Intelligent Autonomous Systems (IAS)*. Springer.

Nguyen, T. T. L., Edward, B., Ki Myung Brian, L., and Graeme, B. (2023). Adaptive trajectory library planner for fast outdoor robots. In *Proc. of Australian Conference on Robotics and Automation (ACRA)*.

Palmieri, L., Koenig, S., and Arras, K. O. (2016). Rrt-based nonholonomic motion planning using any-angle path biasing. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2775–2781.

Park, M. G., Jeon, J. H., and Lee, M. C. (2001). Obstacle avoidance for mobile robots using artificial potential field approach with simulated annealing. In *ISIE 2001. 2001 IEEE International Symposium on Industrial Electronics Proceedings (Cat. No.01TH8570)*, volume 3, pages 1530–1535 vol.3.

Patel, U., Kumar, N. K. S., Sathyamoorthy, A. J., and Manocha, D. (2021). Dwa-rl: Dynamically feasible deep reinforcement learning policy for robot navigation among mobile obstacles. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6057–6063.

Peng, J., Chen, Y., Duan, Y., Zhang, Y., Ji, J., and Zhang, Y. (2021). Towards an online rrt-based path planning algorithm for ackermann-steering vehicles. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, page 7407–7413. IEEE Press.

Peng, M., Gong, Z., Sun, C., Chen, L., and Cao, D. (2020). Imitative reinforcement learning fusing vision and pure pursuit for self-driving. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3298–3304.

Pivtoraiko, M. and Kelly, A. (2005). Efficient constrained path planning via search in state lattices. In *Proceedings of 8th International Symposium on Artificial Intelligence, Robotics and Automation in Space (iSAIRAS '05)*.

Qin, H., Shao, S., Wang, T., Yu, X., Jiang, Y., and Cao, Z. (2023). Review of autonomous path planning algorithms for mobile robots. *Drones*, 7(3).

Quinlan, S. and Khatib, O. (1993). Elastic bands: connecting path planning and control. In *[1993] Proceedings IEEE International Conference on Robotics and Automation*, pages 802–807 vol.2.

Rösmann, C., Feiten, W., Wösch, T., Hoffmann, F., and Bertram, T. (2012). Trajectory modification considering dynamic constraints of autonomous robots. In *Proc. of German Conference on Robotics*.

Rösmann, C., Hoffmann, F., and Bertram, T. (2017). Integrated online trajectory planning and optimization in distinctive topologies. *Robotics and Autonomous Systems*, 88:142–153.

Rösmann, C. (2023). teb_local_planner. `https://wiki.ros.org/teb_local_planner`.

Sánchez, M., Morales, J., Martínez, J. L., Fernández-Lozano, J., and García-Cerezo, A. (2022). Automatically annotated dataset of a ground mobile robot in natural environments via gazebo simulations. *Sensors*, 22(15):5599.

sang Liu, L., feng Lin, J., Yao, J., He, D., Zheng, J., Huang, J., and Shi, P. (2021). Path planning for smart car based on dijkstra algorithm and dynamic window approach. *Wirel. Commun. Mob. Comput.*, 2021:8881684:1–8881684:12.

Schulman, J., Ho, J., Lee, A. X., Awwal, I., Bradlow, H., and Abbeel, P. (2013). Finding locally optimal, collision-free trajectories with sequential convex optimization. *Robotics: Science and Systems IX*.

Shin, Y. and Kim, E. (2021). Hybrid path planning using positioning risk and artificial potential fields. *Aerospace Science and Technology*, 112:106640.

Shu-Xi, W. (2012). The improved dijkstra's shortest path algorithm and its application. *Procedia Engineering*, 29:1186–1190. 2012 International Workshop on Information and Electronics Engineering.

Singh, Y., Sharma, S., Sutton, R., Hatton, D., and Khan, A. (2018). A constrained a* approach towards optimal path planning for an unmanned surface vehicle in a maritime environment containing dynamic obstacles and ocean currents. *Ocean Engineering*, 169:187–201.

Smith, J. S., Xu, R., and Vela, P. (2020). egoteb: Egocentric, perception space navigation using timed-elastic-bands. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2703–2709.

Souissi, O., Benatitallah, R., Duvivier, D., Artiba, A., Belanger, N., and Feyzeau, P. (2013). Path planning: A 2013 survey. In *Proceedings of 2013 International Conference on Industrial Engineering and Systems Management (IESM)*, pages 1–8.

Stentz, A. (1994). Optimal and efficient path planning for partially-known environments. In *Proc. of IEEE International Conference on Robotics and Automation (ICRA)*, pages 3310–3317.

Sukhil, V. and Behl, M. (2021). Adaptive lookahead pure-pursuit for autonomous racing.

Sun, X., Koenig, S., and Yeoh, W. (2008). Generalized adaptive a*. In *Proceedings of the 7th International Joint Conference on Autonomous Agents and Multiagent Systems - Volume 1*, AAMAS '08, page 469–476, Richland, SC. International Foundation for Autonomous Agents and Multiagent Systems.

Sun, Y., Fang, M., and Su, Y. (2021). Agv path planning based on improved dijkstra algorithm. *Journal of Physics: Conference Series*, 1746.

Thrun, S. (2003). Learning occupancy grid maps with forward sensor models. *Autonomous Robots*, 15:111–127.

Viswanathan, V. K., Dexheimer, E., Li, G., Loianno, G., Kaess, M., and Scherer, S. (2020). Efficient trajectory library filtering for quadrotor flight in unknown environments. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2510–2517.

Wang, W., Ru, L., Lu, B., and Hu, S. (2023a). Path planning of uav crossing dense obstacle area based on improved dynamic window approach. In *2023 5th International Conference on Electronic Engineering and Informatics (EEI)*, pages 468–473.

Wang, W.-J., Hsu, T.-M., and Wu, T.-S. (2017). The improved pure pursuit algorithm for autonomous driving advanced system. In *2017 IEEE 10th International Workshop on Computational Intelligence and Applications (IWCIA)*, pages 33–38.

Wang, X., Cheng, M., Zhang, S., and Gong, H. (2023b). Multi-uav cooperative obstacle avoidance of 3d vector field histogram plus and dynamic window approach. *Drones*, 7(8).

Warren, C. (1990). Multiple robot path coordination using artificial potential fields. In *Proceedings., IEEE International Conference on Robotics and Automation*, pages 500–505 vol.1.

Warren, C. (1993). Fast path planning using modified a* method. In *[1993] Proceedings IEEE International Conference on Robotics and Automation*, pages 662–667 vol.2.

Wu, J., Ma, X., Peng, T., and Wang, H. (2021). An improved timed elastic band (teb) algorithm of autonomous ground vehicle (agv) in complex environment. *Sensors*, 21(24).

Wu, J., Ren, H., Lin, T., Yao, Y., Fang, Z., and Liu, C. (2023). A pure electric driverless crawler construction machinery walking method based on the fusion slam and improved pure pursuit algorithms. *Sensors*, 23(18).

Yan, X., Ding, R., Luo, Q., Ju, C., and Wu, D. (2022). A dynamic path planning algorithm based on the improved dwa algorithm. In *2022 Global Reliability and Prognostics and Health Management (PHM-Yantai)*, pages 1–7.

Yasuda, S., Kumagai, T., and Yoshida, H. (2023). Safe and efficient dynamic window approach for differential mobile robots with stochastic dynamics using deterministic sampling. *IEEE Robotics and Automation Letters*, 8(5):2614–2621.

You, A., Sukkar, F., Fitch, R., Karkee, M., and Davidson, J. R. (2020). An efficient planning and control framework for pruning fruit trees. In *Proc. of IEEE International Conference on Robotics and Automation (ICRA)*, pages 3930–3936.

Yuan, H., Li, H., Zhang, Y., Du, S., Yu, L., and Wang, X. (2022). Comparison and improvement of local planners on ros for narrow passages. In *2022 International Conference on High Performance Big Data and Intelligent Systems (HDIS)*, pages 125–130.

Yuan, Q., Yi, J., Sun, R., and Bai, H. (2021). Path planning of a mechanical arm based on an improved artificial potential field and a rapid expansion random tree hybrid algorithm. *Algorithms*, 14(11).

Zhang, M., Shen, Y., Wang, Q., and Wang, Y. (2010). Dynamic artificial potential field based multi-robot formation control. In *2010 IEEE Instrumentation Measurement Technology Conference Proceedings*, pages 1530–1534.

Zhang, Y., Yang, J., Ponce, J., and Kong, H. (2018). Dijkstra model for stereo-vision based road detection: A non-parametric method. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5986–5993.

Zhu, D.-D. and Sun, J.-Q. (2021). A new algorithm based on dijkstra for vehicle path planning considering intersection attribute. *IEEE Access*, 9:19761–19775.

Zucker, M., Ratliff, N., Dragan, A. D., Pivtoraiko, M., Klingensmith, M., Dellin, C. M., Bagnell, J. A., and Srinivasa, S. S. (2013). Chomp: Covariant hamiltonian optimization for motion planning. *The International Journal of Robotics Research*, 32(9-10):1164–1193.