MODEL-BASED REINFORCEMENT LEARNING FOR TRUCK TRAILER ROBOTICS VEHICLE'S PATH PLANNING AND CONTROL

A dissertation proposal in partial fulfillment

of the requirements for the degree of

Doctor of Philosophy

in

Electrical and Computer Engineering

October, 23 2023

By

Hao Yan

Dissertation Committee:

Mohamed A. Zohdy

Amanpreet Kaur

Li Li

Ryan Monroe

Depart of Electrical and Computer Science

ABSTRACT

Advanced Driver Assistance Systems (ADAS) are gaining more attention in the automotive industry during the past decades, driven by technological advancements, consumer demand for safety, and regulatory support, and the ultimate goal of autonomous vehicles is to improve the traffic system to achieve higher efficiency, fewer accidents, better energy consumption. In recent years, advancements in sensor technology, hardware, and innovative algorithms have brought the prospect of self-driving vehicles closer to reality, and the market statistics also show consistent growth in ADAS-equipped vehicles, reflecting the industry's commitment to innovation and safety. Simultaneously, there is an escalating need in the transportation industry to enhance efficiency and minimize the environmental footprint associated with the movement of goods and individuals. As a result, numerous prominent companies in the automotive and technology sectors are now focusing their efforts on the creation and refinement of advanced driver assistance systems and autonomous vehicles.

To mitigate both legal and functional challenges, it is anticipated that high driving automation will first find its application in environments that are both controlled and predictable such as mining fields, harbors and ports, distribution centers and warehouses, agricultural landscapes, and the domain of urban last-mile delivery. Companies like Caterpillar Inc. and John Deere have already marketed autonomous vehicles for mining, farming, and other applications, with collaborations including Carnegie Mellon's National Robotics Engineering Center [1].

In both commercial and industrial sectors, trailer vehicles have seen substantial growth in popularity, becoming vital in transportation, recreation, and commerce. This trend results from various socioeconomic factors, technological advancements, and regulatory shifts. Statistics from the Trailer Output Report show the global light car trailer market

size was valued at USD 1.55 billion in 2021 and is expected to expand at a compound annual growth rate (CAGR) of 3.7% from 2022 to 2030. Smith et al. (2018) also emphasizes the rise of trailers for recreational purposes. However, the varying weights of trailers significantly alter single-vehicle dynamics, creating instability modes like jackknifing and snaking. Therefore, this research aims to achieve efficient planning and control techniques for truck-trailer wheel robot systems using deep neural networks.

Over the past decades, extensive research efforts have been dedicated to overcoming the intricate challenges associated with TTWR (Truck-Trailer Wheeled Robot) systems. This has resulted in the development of various control algorithms, including PID (Proportional-Integral-Derivative), MPC (Model Predictive Control), fuzzy logic, and other nonlinear control methods, each tailored to address specific complexities within TTWR dynamics. Path planning techniques such as Dubins paths, Reeds-Shepp paths, the A* algorithm, Bezier Curves, and Rapidly-Exploring Random Trees have also been applied to TTWR systems. These have been integrated with different control algorithms to ensure closed-loop stability, with the effectiveness of the proposed solutions being validated through simulations. However, the classical path planner and controller is limited with the tuning Complexity and Lack of Adaptation, which limits the user scenario of TTWR assistance

This research contributes an end-to-end deep reinforcement learning network solution for TTWR reverse autonomous control in low-speed, closed, and unstructured environments, alongside a comparison of classic control methods. The research reviews and contrasts popular motion planning and feedback control frameworks, guaranteeing closed-loop stability and validating proposed solutions via simulations. Path planning comparisons were made among A-star search, Dubins path, and polynomial path, including strategies like linear quadratic (LQ) control and also advanced model predictive control (MPC) techniques to account for physical and sensing limitations.

The primary goal of the end-to-end DRL controller was to achieve TTWR autonomous

reverse driving with a light model based deep reinforcement learning network suitable for deployment on embedded automotive platforms. Although several end-to-end deep neural networks exist for autonomous driving, where the input to the machine learning algorithm are camera images and the output is the steering angle prediction, but traditional neural networks are less robust compared with deep reinforcement learning based control algorithm. Furthermore, the Model-Based DRL potentially improve sample efficiency by allowing an agent to synthesize large amounts of imagined experience. To demonstrate the efficiency, several DRL algorithms are developed with different computational complexity and performance, and these algorithms replace different classical control algorithm units including path planning, control, and then the end-to-end DRL controller is proposed which takes raw sensor data as input, and then generate steering and throttle signal to control the TTWR movement and achieve target chasing, obstacle avoidance and ensurance of passenger comfort at the same time.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 1

Introduction

Advanced driver assistance systems (ADAS) have gained significant attention as a revolutionary technology for improving travel experiences, aiming to enhance safety, providing active assistance to drivers, and facilitating the development of fully autonomous driving technology. To handle the complex road geometry and topology, multi-agent interactions, and accurately follow the high-level command such as routing information, the machine learning algorithms have been widely integrated for ADAS and are becoming popular in this decades [2] [3].

Figure 1.1: Truck trailer system in daily life

Among the various features of ADAS, the control and auto-parking assist systems for

tractor-trailer wheeled robots (TTWR) have gained significant attention in recent years, because of trailer vehicles' ability to significantly increase load capacity at a relatively low cost, resulting in more efficient use of resources and reduced transportation costs for achieving cooperative tasks[4]. However, the nonlinear, nonholonomic, unstable, and under-actuated system behaviors, along with the large blind zone behind the trailer make TTWR reverse parking a challenging and stressful experience. Another difficult part of the system is the phenomenon called jackknife, which occurs when the angle between the tractor and trailer is beyond a certain threshold, which prevents the driver from steering or aligning the trailer properly during driving backward. When the TTWR system is in the jackknife state, the steering maneuver has little effect on controlling the trailer's states.

Extensive research, for decades now, has been dedicated to addressing the challenges associated with TTWR systems, resulting in the development of various controlling methods that use controllers such as fuzzy logic, neural networks, and nonlinear controllers. For example, traditional control techniques, including Proportional-Integral-Derivative (PID) and model predictive controllers, have been widely utilized in Jackknife prevention, as well as other systems such as steer-by-knob. Jing et. al [5] solved the tractor-trailer reverse motion by designing three distinct controllers using a proportional integral controller, a sliding mode controller, and a neural network controller respectively. Moreover, a generic control safety governor was developed to supervises the tracking algorithms, overriding control to ensure jackknife-free operation. Xu et. al [6] revisited Proximal Policy Optimization (PPO), enhancing it through two main innovations: firstly, they reformulate PPO into a linearly combined form to control the trade-off between accumulative discount return and divergence, addressing parameter tuning complexities, and secondly, they introduce a parametric alpha divergence in place of the traditional Kullback–Leibler (KL) divergence for more effective policy differentiation. The novel variant, alphaPPO, is validated across six benchmark environments, outperforming existing PPO forms. The

work contributes a more effective balance in return and divergence, and a refined divergence measurement, broadening PPO's application in reinforcement learning. Zanchetta et. al [7] propose a method to control the trailer's lateral position through vehicle yaw moment control, and by employing a control allocation algorithm and considering various constraints such as actuator limits and tire saturation, they present a robust solution for trailer control. This study extensively uses simulations, validated with real-world experimental data, to demonstrate the effectiveness of the method. This research contributes to the field of autonomous driving, specifically enhancing trailer control, and offers a novel approach that could be applicable to various vehicle configurations and driving conditions.

As technology evolves, especially in recent years, the application of machine learning algorithms for ADAS has emerged as a key element in the automotive industry for autonomous driving, as well as driver comfort and safety. Deep reinforcement learning has shown great potential for handling decision-making and controlling problems [8]. It has become a popular research topic, in the field of continuous control development, for its flexibility and potential for improved control performance through trial and error. Unlike traditional supervised learning methods, deep reinforcement learning agents learn control policies through interactions within that environment, making them suitable for dynamic and unpredictable environments that are encountered in various applications such as autonomous driving and robotics [9]. Several ADAS problems have been tackled using reinforcement learning algorithms [10][11][12], but the development of end-to-end autonomous reverse parking systems for TTWR is still an ongoing area of investigation and growth. To overcome the limitations of rule-based models, which are prone to failures in unknown environments, Du et. al [11] present a novel trajectory planning method for automated parking systems using Deep Q-Network (DQN) learning, they apply the DQN algorithm to generate optimal paths in complex parking scenarios, considering the vehicle's nonholonomic constraints. Wang et. al [13] propose a policy gradient reinforcement learning method specifically for reverse motion control of tractor-trailer mobile

robots. Utilizing a continuous action space, the authors focus on the backtracking control problem and the challenge of jackknifing prevention, they employ a deep deterministic policy gradient (DDPG) algorithm to generate continuous control actions, thus providing an innovative approach to the problem. Through extensive simulations and comparison with other methods, the paper validates the effectiveness of the proposed approach in enhancing stability and safety during reverse driving of tractor-trailer systems. Bejar et.al [14] developed a preview neuro-fuzzy controller based on deep reinforcement learning for truck-trailer vehicle systems, and the research method includes utilizing a deep deterministic policy gradient algorithm integrated with a fuzzy logic rule-based system to train the controller. By combining deep reinforcement learning with neuro-fuzzy control, the safety performance and jackknifing avoidance are significantly improved. The paper demonstrates the application of modern machine learning techniques to a complex control problem, providing a new perspective on trailer vehicle control.

Model-based deep reinforcement learning (MBDRL) combines the benefits of deep learning with model-based control techniques, offering a novel approach for vehicle control. Unlike model-free methods that requires extensive interactions with the environment, MBDRL achieves proficient policies with fewer samples by utilizing a learned model of the environment. This model aids in simulating potential future outcomes, enabling the algorithm to anticipate and make informed decisions. While this "thinking ahead" capability enhances efficiency, it's essential to ensure the model's accuracy, as any discrepancies between the learned model and the real-world environment can lead to suboptimal or even erroneous control actions.

To conclude, the truck trailer wheeled robot systems present unique challenges compared to traditional cars and trucks. Their larger size and inherent instability during reverse movements make the task of crafting efficient motion planning and feedback control mechanisms especially complex, especially in the context of autonomous reversing. Modern solutions using deep reinforcement learning and neural networks offer innovative

ways to enhance trailer movement performance. Leveraging AI-based techniques, these approaches can learn from complex driving scenarios and adapt to unforeseen conditions, thereby improving safety. Furthermore, the implementation of end-to-end parking solutions, driven by artificial intelligence, can significantly enhance user experience and efficiency by automating intricate parking maneuvers and allowing for a more seamless integration with existing transportation infrastructures. The combination of classical control algorithm and intelligent algorithms is opening new possibilities for handling these intricate control challenges, enabling more robust and responsive systems.

Chapter 2

Motivation and Research Objective

The research objective of the truck trailer wheel robot (TTWR) system is to harness the potential of model-based deep reinforcement learning to achieve end-to-end control, with a focus on enhancing efficiency, stability, and real-world applicability. The main goals and key components of this research include:

1. Building Trailer Movement State Equations: Developing precise mathematical models to represent the state dynamics of the TTWR system. This foundational work will create a framework for subsequent control and planning algorithms.

2. Comparison of Classical Path Planning Methods: Investigating and comparing different classical path planning methods, including:

    - Polynomial curve: One of the most simple and popular methods used in nowadays industrial and research field due to its computational efficiency and robust performance, it also provides a flexible framework that can be easily adapted to various requirements

    - Clothoid curve: Providing smooth transitions between curves of different curvatures, which property makes it an ideal choice for applications such as road design, robotics, and autonomous vehicle navigation, where smooth and continuous paths are often desired.

- Dubins curve: Exploring its applicability for smooth path generation. The popularity of this algorithm is because of its simplicity and analytical tractability, and the algorithm also considers the physical constraints of the vehicle, such as its minimum turning radius.
- A* search: Analyzing its efficiency in finding optimal paths in complex environments. The A* search is an extension of Dijkstra's algorithm, with the added capability of using a heuristic to estimate the cost from the current node to the goal, allowing it to find optimal paths more efficiently.

3. Comparison of Different Control Algorithms: Conducting a comparative study of different control algorithms to identify the most effective approach for the TTWR system, including:

    - PID: Evaluating its robustness and simplicity. Although PID controllers are simple, well-understood, and widely used in various industries,and hey are computationally efficient and work well for systems with known and stable dynamics, the tuning of PID parameters can be challenging, especially for complex or nonlinear systems. PID controllers may not handle disturbances or model uncertainties well, and they may require manual re-tuning if system dynamics change.
    - LQR: Assessing its ability to handle nonlinear TTWR systems. Because LQR is mainly optimal for linear systems, providing robust performance by minimizing a quadratic cost function. It can handle multi-input and multi-output systems and can be designed to meet specific performance criteria. But when for nonlinear model, the lionization will cause inaccuracy because the LQR algorithm requires a precise mathematical model of the system and assumes linearity, moreover, designing and solving the underlying optimization problem can be computationally intensive.

- MPC: Analyzing its potential to handle constraints and model nonlinearities. The advantage of MPC is that the consideration of future states by solving an optimization problem at each step, allowing it to handle constraints and adapt to changes in system dynamics. But it requires a detailed mathematical model and can be computationally demanding, especially for high-frequency or real-time applications. The optimization problem must be solved at each time step, which may lead to challenges in implementation and tuning.

4. Implementation of Model-Free Reinforcement Learning: Implementing and evaluating different model-free reinforcement learning algorithms, such as:

    - DQN: Investigating its effectiveness in discrete action spaces. DQN combines Q-learning with deep neural networks, enabling the handling of high-dimensional input spaces. It's robust and has been successfully applied to various tasks, especially in discrete action spaces. However, the continuous action spaces and instability during will be a problem in real world scenario training. The experience replay buffer required can be memory-intensive.
    - DDPG: DDPG extends the ideas of DQN to continuous action spaces by combining Q-learning with policy gradients. It has shown success in various complex environments. But DDPG can be sensitive to hyperparameters and might suffer from training instability. It requires careful tuning and normalization techniques to achieve optimal performance.
    - TD3: Assessing its ability to achieve stable learning in complex environments. TD3 improves upon DDPG by addressing overestimation bias and instability. It introduces a clipped double-Q learning and delayed policy updates, leading to more stable and efficient training. TD3, while more stable than DDPG, still requires careful hyperparameter tuning. The additional complexity in the algorithm might increase the implementation difficulty compared to simpler

methods like DQN.

5. Implementation of Model-Based Reinforcement Learning: Implementing and evaluating different model-based reinforcement learning algorithms, such as:

   - MBRL with known model [15]: where we plan over a known model, and only use learning for the global solution. The learnt model is based on classical trajectory planning including A*, RRT, or Dubins path, and after integrated with TTWR physical model, the model will generate TTWR trajectory states for the controller to take as input. The model-free RL algorithm will be used for the agent to interact with the environment, which is shown in figure 2.1
   - Model-based RL with a learned model, where we both learn a model and learn a global solution.

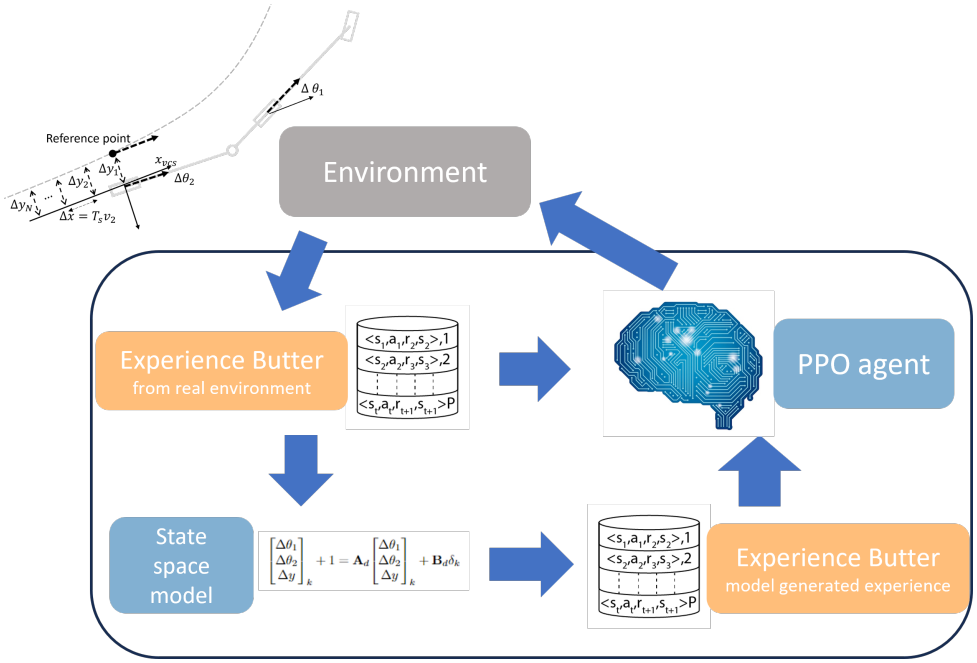


Figure 2.1: MBRL based on known model

The proposed research aims to introduce a noval approach using model-based reinforcement learning for end-to-end control in truck-trailer wheel robot systems. While this is an exciting direction, it comes with its own challenges:

1. World State Representation: Effectively representing the world states, including surrounding objects, system states, and movement cost, is a complex task that requires innovative methods to capture all relevant information without overwhelming the computational resources.

2. Stability of Reinforcement Learning Algorithm: Ensuring the stability of the reinforcement learning algorithm, especially in the face of non-stationary and unpredictable real-world scenarios, is a critical challenge. Fine-tuning the balance between exploration and exploitation and devising robust training techniques will be essential.

3. Computational Efficiency in Complex Environments: Designing computationally efficient methods that can be deployed in complex real-world environments without sacrificing performance or safety is a significant hurdle. Developing lightweight models and optimizing computation will be key to the successful implementation of the TTWR system.

This research aims to make significant improvements in the control and automation of TTWR systems. By tackling the challenges and focusing on modern AI-based solutions, the goal is to enhance safety, efficiency, and adaptability in autonomous vehicles and complex robotic systems. The success of this work could lead to practical applications that are more responsive and reliable, improving efficiency of daily transportation and optimize energy usage.

Chapter 3

Research background

In the past decade, advancements in sensor and hardware capabilities have brought the concept of autonomous vehicles closer to realization. The popularity of events including the DARPA Grand Challenges [16][17] has motivated major automotive companies and tech companies to further develop self-driving technology and deploy such technologies in daily life scenario. The potential benefits of eliminating human intervention in driving are manifold, which aims to enhance road safety, reduce traffic congestion, decrease carbon emissions, and offer mobility to those unable to drive. It represents a shift towards sustainable and efficient transportation, potentially transforming urban planning, energy consumption, and societal accessibility. Such technological strides not only offer car companies an opportunity to differentiate themselves in a competitive market but also align with the increasing demands of the transportation sector. With transportation being a significant contributor to global carbon emissions, regulatory bodies such as the European Road Transport Research Advisory Council and the European Commission have delineated ambitious targets. Their vision is to achieve a 50% enhancement in transport efficiency by 2030, relative to 2010, and simultaneously curtail emissions by 60%.

Nowadays, Advanced Driver Assistance System (ADAS) has been developed to enhance road safety by aiding drivers in maintaining vehicle stability, and has emerged as a bridge between today's vehicles and the future of full autonomy. While classical perception, path planning, and motion control methods can handle the majority of driving

scenarios, there remain certain corner cases where these traditional techniques fails. These complex and often unpredictable scenarios present unique challenges that cannot be easily resolved using conventional approaches. As a result, the modern ADAS algorithm integrates various sensors and algorithms to assist drivers, making driving safer and more efficient. From simple parking assistance to complex adaptive cruise control, these systems represent the intersection of robotics, artificial intelligence, and vehicular engineering and try to create more sustainable, efficient, and safer transportation solutions.

Compared with vehicles which does not have additonal trailer connected to the ego vehicle body, the truck-trailer wheeled robot, have always brought challenges to the classical state estimation and control methods. While there has been significant progress in developing state and parameter estimation techniques for single-unit vehicles, the area of vehicle-trailer system state and parameter estimation still can be explored, and especially when considering the task of reverse driving. The dynamics involved in reversing a truck-trailer system differ significantly from those of forward driving, primarily due to the inherent instability and non-holonomic constraints (constraints that are non-integrable into positional constraints) of the system. When a truck attempts to reverse with a trailer, the angle between the truck and trailer can quickly grow uncontrollable, leading to a phenomenon known as jack-knifing.

In the previous research, control of such systems was the domain of skilled human operators; however, with the advent of automation and the push towards autonomous vehicles to wider user scenario, there has been a growing interest in developing algorithms that can reliably and safely control truck-trailer systems during reverse driving. Traditional control methods, such as Proportional-Integral-Derivative (PID) controllers, Linear Quadratic Regulators (LQR), and Model Predictive Control (MPC), have been explored for this purpose. While these methods have seen success, they often rely on accurate models of the system dynamics and can struggle with the nonlinearities and uncertainties from the real-world driving scenarios. In recent times, methods that take advantage of

modern machine learning techniques, such as deep reinforcement learning , have been introduced, which combines the capabilities of deep learning for recognizing complex patterns with reinforcement learning for decision-making. This fusion enables systems to learn and adapt through continuous interaction with complex environment and adjust its strategy to achieve optimal control.

3.1 Overview of path planning

Historically, path planning was primarily based on geometric and grid-based methods. Geometric methods, such as the Voronoi diagram, partitioned the workspace into regions to find the safest path. Grid-based methods, on the other hand, divided the environment into a grid and used search algorithms to find the path. However, these methods often faced challenges in complex environments or when high precision was required. One of the most popular methods used in autonomous parking is Dubins curve, which is named after Lester Eli Dubins, is a mathematical solution that provides the shortest curve between two points in the plane with a constraint on the curvature. While the Dubins Curve itself was introduced in the 1950s, and with an assumption that the vehicle traveling the path can only travel forward. If the vehicle can also travel in reverse, then the path follows the Reeds–Shepp curve [18], its application to automated parking has been a more recent development, and its integration into parking algorithms allowed for smooth and efficient paths that adhered to the nonholonomic constraints of vehicles, such as a fixed minimum turning radius.

Nowadays, graph search algorithms are becoming dominantly popular and are fundamental to path planning, as they provide systematic ways to explore and navigate through a network of nodes and edges representing the environment. The most basic graph search algorith is Dijkstra algorithm [19], and it has been a foundational method commonly used in applications like Google Maps, known for its guaranteed optimal solutions but criticized

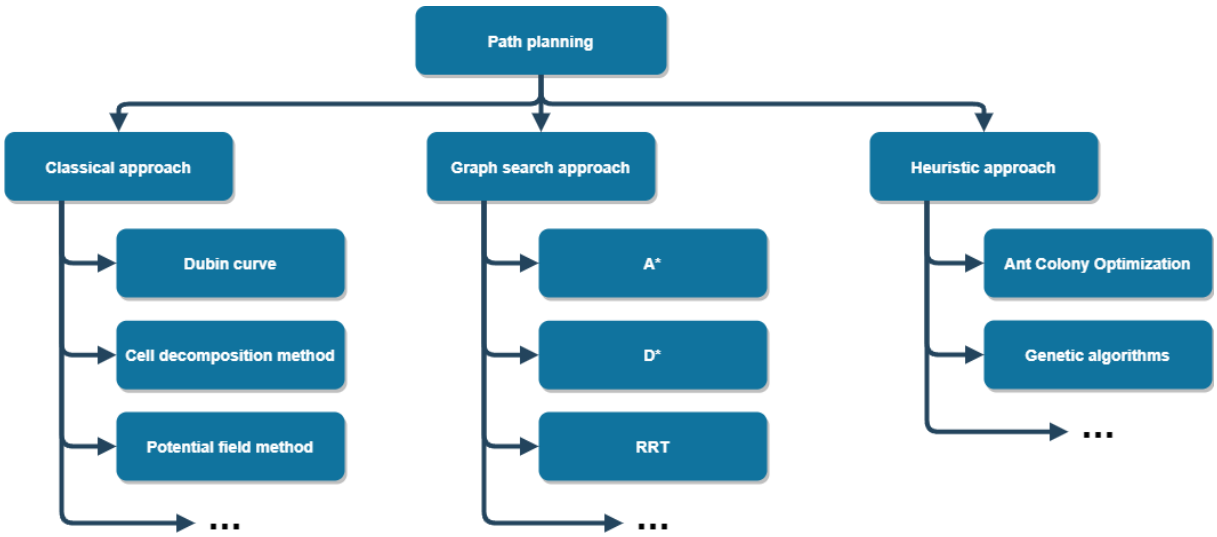


Figure 3.1: Map of path planning methods

for its computational intensity in blind searches. However, the computational-intensity limits the Dijkstra's blind searches capability, and as a result A* [20] and its variants are becoming the state of the art algorithms for use within static environments, they combined the benefits of both, using a heuristic to guide the search towards the goal while also considering the actual cost from the start. A* guarantees to find the shortest path if an admissible and consistent heuristic is used, and by using a heuristic that estimates the cost to the goal, A* often explores fewer nodes than algorithms like Dijkstra's, making it more efficient in many cases. The A* algorithm and its variants are a significant milestone in the field of path planning, because the balance between efficiency and optimal helped the industry to reduce cost and improve stability.

However, A* and A* re-planner are used for shortest path evaluation based on the information regarding the obstacles present in the environment, and the shortest path evaluation for the known static environment is a two-level problem, which comprises a selection of feasible node pairs and shortest path evaluation based on the obtained feasible node pairs [21]. Both of the above mentioned criteria are not available in a dynamic environment, which makes the algorithm inefficient and impractical in dynamic environments. To support path planning in dynamic environments, D* [22] and its variants are

discussed as efficient tools for quick re-planning in cluttered environments. As D* and its variants do not guarantee solution quality in large dynamic environments, we also explore Rapidly-exploring Random Trees (RRTs) [23] and a hybrid approach combining Relaxed A* (RA*) [24] and one meta-heuristic algorithm. The hybrid approach comprises two phases: the initialization of the algorithm using RA* and a post-optimization phase using one heuristic method that improves the quality of solution found in the previous phase. Three meta-heuristic algorithms are also discussed: namely, the Genetic, Ant colony, and Firefly algorithms. These are aimed at providing effective features in pursuit of a hybrid approach to path planning.

The A* algorithm, although effective in static environments, its application in dynamic settings is hindered by its two-level problem structure, involving the selection of feasible node pairs and path evaluation based on these pairs. This makes A* less practical in ever-changing environments. To address the challenges in dynamic environments, incremental versions of A* such as the D* algorithm have been developed, allowing for path updates as new information becomes available1. However, D* and its derivatives may not always ensure solution quality in extensive dynamic landscapes [22]. D* is designed to handle changes in the environment, allowing for efficient replanning without having to restart the search from scratch, and it is able to reuse the previous computations, and efficiently update the solution when changes occur, making it suitable for real-time applications. However, D* and its derivatives may not always ensure solution quality in extensive dynamic landscapes, and as result, A rapidly exploring random tree (RRT) is designed to efficiently search nonconvex, high-dimensional spaces by randomly building a space-filling tree.

Path planning is a critical aspect of autonomous driving and ADAS, guiding the ego vehicle go through complex environments. Algorithms like Dubins curve, Dijkstra's, A*, RRT, and D* are chosen as they represent different facets of path planning, serving as various scenarios from static to dynamic environments, and from deterministic to proba-

bilistic approaches. The trend of integrating AI, particularly machine learning, into path planning is reshaping the field. By leveraging data-driven insights and adaptive algorithms, AI-enhanced path planning offers the potential for more robust, efficient, and intelligent navigation. This fusion of traditional algorithms with AI techniques signifies a promising direction for future innovation and optimization in path planning [25].

3.2 Overview of vehicle control

The evolution of control theory in autonomous driving and Advanced Driver Assistance Systems (ADAS) has been marked by the development and application of various control strategies, reflecting the complexity and diversity of the challenges in this field.

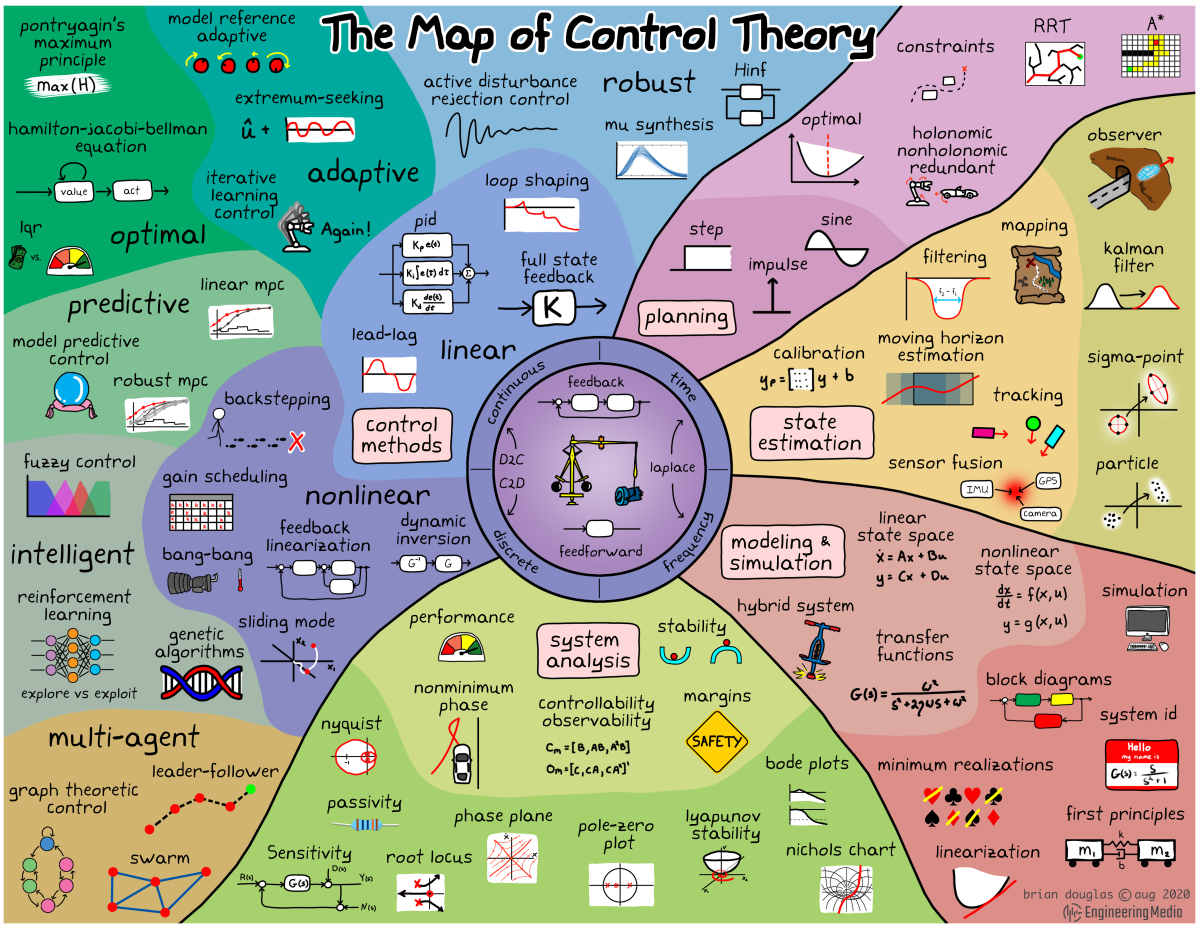


Figure 3.2: Map of control theory[26]

Starting with PID control, which originated in the early 1920s, initially applied to automatic steering systems for ships. This marked the first theoretical analysis and practical utilization of PID in a real-world context. As the technology matured, it found its way into the manufacturing industry, where it played a vital role in automatic process control. Initially implemented in pneumatic controllers, it later transitioned to electronic formats. In the modern era, the PID concept has become a universal standard, employed across various fields and applications where precise and optimized automatic control is essential. Utilizing feedback control, PID controllers adjust actions based on the error between desired and actual states, making them suitable for tracking target velocity and lateral positioning in autonomous driving [27].

The main drawback of PID controllers is that every test on the actual system requires its linearization. For LQR control, this step is not needed, and one can directly feed in the system equations to the controller and get the desired response [28]. As one of the optimal control theory which aims to operate a dynamic system at the lowest cost, the LQR controll can be appplied in the Linear Quadratic (LQ) problem, where system dynamics are defined by linear equations, and the cost is a quadratic function. By minimizing a cost function defined by deviations from desired values, like lateral distance or longitudinal velocity, the controller finds the optimal settings that reduce undesired deviations, and the control action's magnitude may also be part of the cost function.

LQR (Linear Quadratic Regulator) is limited by its assumption of a linear system, caused inconvenience for its application in real-world scenarios which are often highly nonlinear. In contrast, MPC (Model Predictive Control) is designed to handle nonlinear systems without the constraints of linearity. It can manage hard constraints and deviations from a linearized operating point, major drawbacks to LQR. Unlike LQR, which optimizes across the entire time horizon, MPC optimizes within a receding time window, frequently computing new solutions. This flexibility allows MPC to adapt to complex, nonlinear environments, making it more suitable for many real-world applications.

Another control method used widely is called fuzzy control, compared with PID control which requries techniques and algorithms to tune the PID gains by a skilled human operator during the application of the controller to a process, fuzzy control uses fuzzy sets and fuzzy rules to model complex systems. Unlike traditional binary logic, fuzzy logic allows for degrees of truth, representing values on a continuum rather than as absolute true or false. Fuzzy controllers translate these fuzzy sets into control actions, making them suitable for handling uncertainty and ambiguity. Because of its ability to handle imprecise information makes it flexible and applicable to various complex systems, it is becoming a popular choice, especially in applications where traditional PID control might struggle. It represents a more intuitive and adaptable control methodology, suitable for a wide range of real-world scenarios.

3.3 Overview of end to end driving

The concept of end-to-end autonomous driving has evolved from traditional hierarchical architectures to more streamlined and integrated approaches. Historically, autonomous driving systems were designed with complex, layered structures, including separate modules for environment perception, path planning, and motion control. The hierarchical scheme, as seen in Carnegie Mellon's BOSS car [29] and Stanford's Junior [30], was prevalent in the early stages of autonomous driving development.

However, the rise of deep learning and reinforcement learning has paved the way for end-to-end methods. These approaches directly map raw sensor data to low-level control commands, bypassing the need for modularized design. NVIDIA's work in 2016 marked a turning point, realizing end-to-end autonomous driving on real-world freeways [31], where an end-to-end deep learning approach to control an autonomous vehicle were demonstrated. Unlike traditional methods that relied on hand-crafted features and complex pipelines, NVIDIA's approach used a Convolutional Neural Network (CNN) to map

raw pixels from a single front-facing camera directly to steering commands.

Comapred with traditional hierarchical architectures, which is complicated and hard to design, end-to-end methods offer a more straightforward structure, reducing the burden of designing complex modules. And the Adaptability of end-to-end approaches allows new scenarios to be learned by interaction and previous experienced to eb generalized at scale, unlike traditional methods prone to error propagation. Also, because all sensor data is directly used and mapped for driving, there's less perception information loss or error propagation, and the algorithm has the ability to find the correlations between different sensors. Nowadays, the imitation learning or reinforcement learning has been introduced and employed in end-to-end driving research, making it possible for vehicles to learn and improve end-to-end driving by themselves.

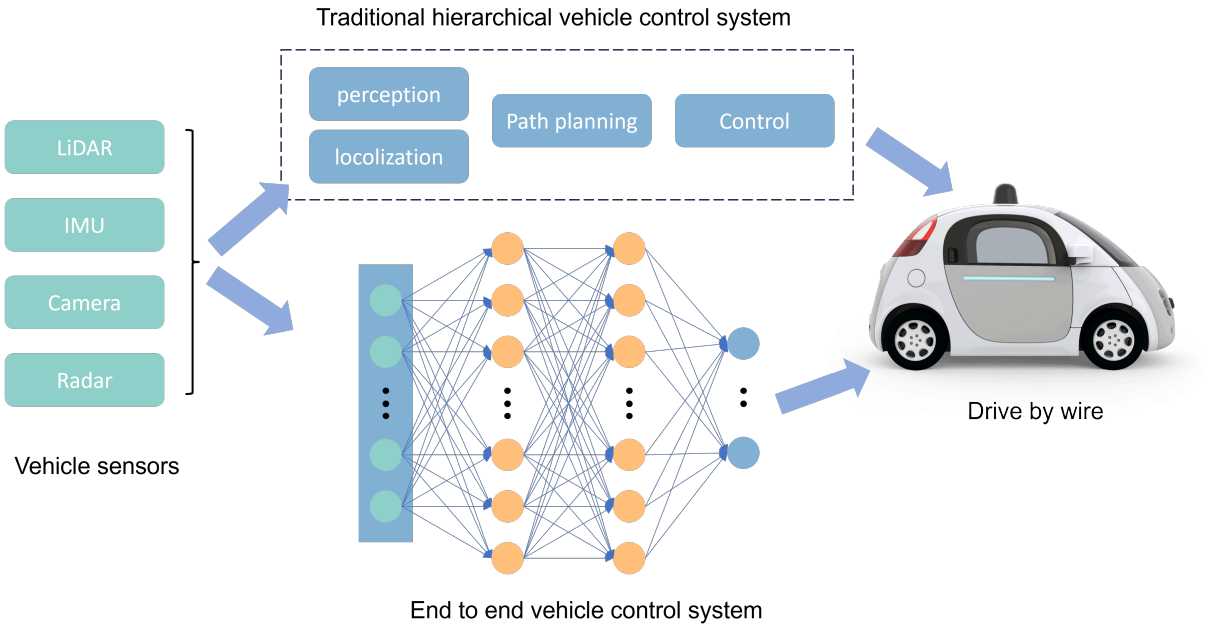


Figure 3.3: Comparison between traditional hierarchical vehicle control system and end to end vehicle control system

In the realm of end-to-end autonomous driving, various research methods have been employed to achieve seamless control and navigation. The work from Nvidia by using a Convolutional Neural Network to directly map raw pixels from cameras to steering commands has been a foundational approach, simplifying the control process by directly

linking visual input to driving actions. Building on this, Imitation Learning (IL), or learning from demonstrations, has emerged as a technique where an agent learns the optimal policy by imitating expert human behavior, providing a data-driven way to capture complex driving strategies. Reinforcement learning has extend this idea further by mapping raw pixels to discretized actions, offering a more interactive and self improving control mechanism, which demonstrated successful results on real world driving recently, including lane following [32] and urban driving [12]. Peng et al [33] introduces an end-to-end autonomous driving approach utilizing the Dueling Double Deep Q-Network, a deep reinforcement learning algorithm. This method enables the vehicle to independently learn end-to-end driving, fostering self-reliance in navigation and control. Actor-Critic Methods, such as A3C, DDPG, and TD3, have been applied to handle complex decision-making and planning problems, integrating both value and policy-based learning for more robust control. Reinforcement learning has been successfully applied in autonomous driving when combined with supervised learning in recent years. The future of RL in autonomous driving faces challenges in transferring findings from simulation to the real world. Simplistic reward functions may encourage risky behavior [34], and designing or learning better functions remains an open problem. Combining RL with world models and enhancing representation learning are key areas for ongoing exploration and development. Also, current RL solutions for autonomous driving will be hindered by high dimensional representation due to computational cost, and research is still on going for the model to learn from and interact with the world in a more data efficient way [35].

In conclusion, the end-to-end autonomous driving using Deep Reinforcement Learning has emerged as a promising direction, offering simplicity, adaptability, and the potential for superior performance. It stands in contrast to traditional hierarchical methods, providing a more integrated and flexible approach. However, challenges related to data dependency, interpretability, sensor integration, and robustness continue to be areas of active research and development. The ongoing exploration of various DRL algorithms,

combined with real-world training and attention to interpretability, is shaping the future of autonomous driving, making it an exciting and dynamic field of study.

3.4 Model based reinforcement learning in autonomous driving

In reinforcement learning context, unlike other machine learning methods, the learning process is pushed forward using the interactions between the agent and the environment, which means the agent learns control mapping function directly from its surroundings without requiring supervision. However, discrimination between the agent and the environment is not always clear, which varies in different application. For instance, in autonomous driving problems, the vehicle's actuator response are often considered part of the environment.

The main difference between model-based and model free reinforcement learning is whether a model of the interactions between the robot and the environment is employed. MFRL algorithm learns the control policy without such model, and instead, it relies on the method called trial-and-error, which directly with the physical system to discern rewards and determine optimal actions. On the other hand, MBRL incorporates a model which captures the environment transition dynamics, and use this model as the foundation for reward determination and action optimization. Consequently, in model-based approaches, policies are fine-tuned using the model, and once optimized, they are then deployed on the actual system. This process is depicted in Figure 3.4, and shows the typical data flow of a model-based reinforcement learning approach.

Nowadays, the model-based reinforcement learning has been proven to be sampling efficiency and wide adaptable in the field of robotics. Huang et. al. investigated the challenge of parametrizing policies for reinforcement learning in highdimensional continuous action spaces, and present a practical model-based RL method leverages the multimodal

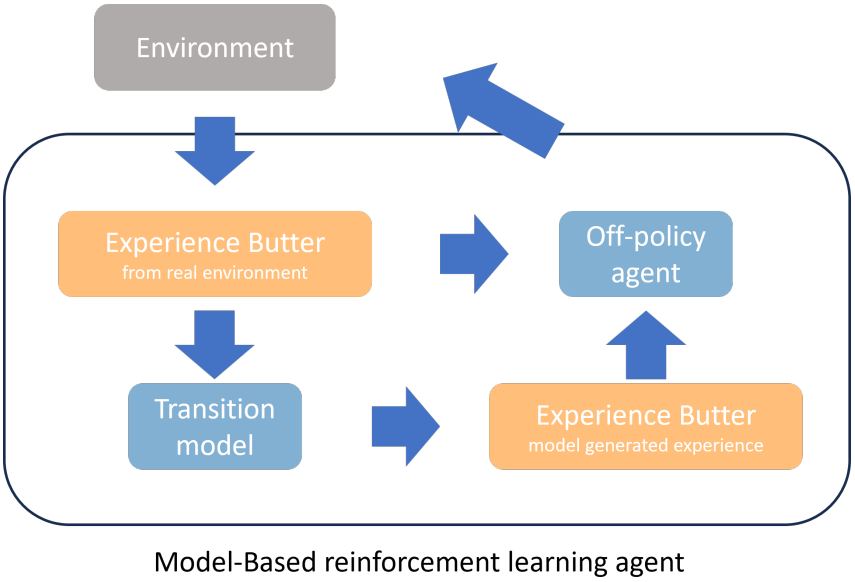


Figure 3.4: Model-based reinforcement learning schema

policy parameterization and learned world model to achieve strong exploration capabilities and high data efficiency [36]. Hu et. al. introduces a model-based reinforcement learning technique for autonomous driving, which learns the world model and driving policy using 3D geometry from high-resolution expert videos. Trained offline on urban driving data, MILE outperforms previous methods by 31% in the CARLA simulator under new conditions. It uniquely integrates static and dynamic scene modeling with ego-behavior using only cameras[37]. Heess et. al. introduced a unified framework for learning continuous control policies via backpropagation, developed multiple RL algorithms based on teh framework, and then compared the general policy gradient algorithms that range from model-free methods with value functions to model-based methods without value functions. The results showed that the model-based algorithm SVG(1) shows the effectiveness of learning models, value functions, and policies simultaneously in continuous domains comapred with other competators.

To conclude, model-based reinforcement learning is becoming a popular research area as it requires fewer interactions with the environment compared to model-free RL, which reduced interaction not only minimizes the risk of accidents but also reduces wear and

tear on the robot. To successfully implement MBRL, the transition models should be designed carefully, which will affect the performance of the learning algorithm in terms of accuracy and convergence. Furthermore, imitation learning combined with RL, by guided by human demonstrations, making RL a promising approach for tasks like autonomous driving where human-like decision-making is preferred.

Chapter 4

TTWR state space analysis

This chapter outlines the main focus of the research, detailing the definition of the TTWR state space and its controllability analysis. The study employs a kinematic truck-trailer vehicle model to represent a simulation vehicle operating at low speeds without tire slip. While a comprehensive model might encompass vehicle dynamics, such as load transfer and tire slip, this research primarily centers on the kinematic aspects for low speed parking scenarios. The chapter concludes by addressing the limitations of the TTWR model and the common methods to avoid such issue.

4.1 TTWR kinematics state space

Previous research [38][39] has provided various physical models for vehicle state analysis, including both kinematic and dynamic state models. The choice of model depends on the desired level of accuracy and the specific operating conditions. For high-speed driving tasks, the dynamic factors play a significant role in motion planning and vehicle control; however, in low-speed driving scenarios, the kinematic constraints have a greater influence on the motion behavior of the TTWR system. Since the TTWR reverse parking task is operated in low-speed driving conditions, the system model is established based on the kinematic relations. In this paper, we utilized the Ackermann chassis vehicle as the host vehicle, and the trailer employed a single-axle chassis for the TTWR system.

Table 4.1: The TTWR system states

Parameters	Descriptions
x_1	The lateral position of the host vehicle
y_1	The longitudinal position of the host vehicle
θ_1	The yaw angle of the trailer
x_2	The lateral position of the trailer
y_2	The longitudinal position of the trailer
θ_2	The yaw angle of the host vehicle
ϕ	The host-trailer angle
v	The longitudinal velocity of host vehicle
δ	The steering angle of host vehicle
L_1	The host vehicle wheelbase length
L_2	The rear overhang of the host vehicle from the rear axle to the hitch point
L_3	The overhang of the trailer from the the hitch to trailer axle

The two vehicles in the system are linked by the hitch ball, which is a unique pivoting point. The pivot point enables the interconnected vehicles to execute relative rotational motion between each other. From a mechanical structure perspective, the tractor and the trailer can be regarded as two separate rigid bodies. The vehicle body structure exhibits symmetry with respect to the axis, with the left and right sides being symmetrical. On flat non-slippery ground and under low-speed driving, the side-slip caused by tires and control effect delay caused by mechanical or electrical units can be neglected[40]. In Table 4.1, the major parameters of the TTWR system are listed.

This state-space model of TTWR in this paper is illustrated in Figure 4.1, where the trailer has a single axle and a conventional overrun braking system. The assumptions for the TTWR system are as summarized as follows:

- The system has a unique pivoting point, known as the hitch, which is essential in yielding the unstable equilibrium point during reverse driving.
- The system is defined on a flat surface and without slide slip in the kinematics model and simulation environment.

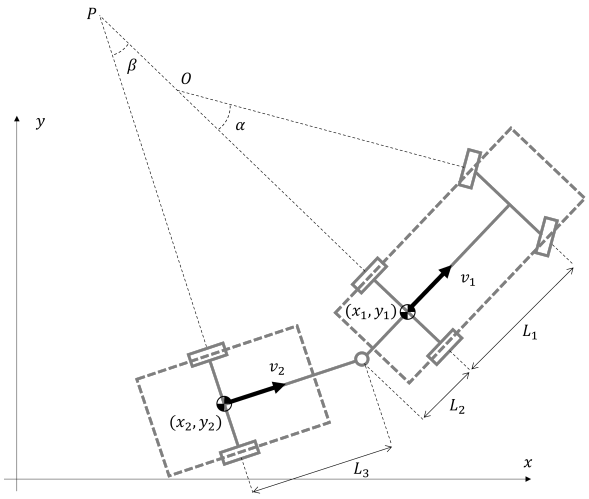


Figure 4.1: Kinematic full-track model of TTWR

- The trailer has a uniformly distributed mass.

Because all the perception values are received in the vehicle coordinate, it is essential to build the reference point coordinate transformation between the vehicle coordinate and the world coordinate. In this paper, only the translation from trailer coordinate to world coordinate is explained, because host coordinate transformation also follows a similar computation.

Under the trailer coordinate, the reference points coordinates can be expressed as:

$$P_2 = \begin{bmatrix} x_{21} & x_{22} & \cdots & x_{2n} \\ y_{21} & y_{22} & \cdots & y_{2n} \end{bmatrix} \tag{4.1}$$

where P_2 is the coordinate vector matrix, each column vector of the matrix represents a reference point (e.g. surrounding obstacle detections, trailer corner coordinate and et al.) under the trailer coordinate system. $(x_{21}, x_{22}, \ldots, x_{2n})$ and $(y_{21}, y_{22}, \ldots, y_{2n})$ are X-coordinate and the Y-coordinate values respectively.

In linear algebra, the rotation matrix R_{world} and the translation matrix T_{world} can be

known as:

$$\mathrm{R}_{\mathrm{world}} = \begin{bmatrix} \cos\theta_2 & -\sin\theta_2 \\ \sin\theta_2 & \cos\theta_2 \end{bmatrix} \tag{4.2}$$

$$\mathrm{T}_{\mathrm{world}} = \begin{bmatrix} t_{x2} & t_{y2} \end{bmatrix}^T \tag{4.3}$$

where the (x_2, y_2) are the origin point of the trailer coordinate on the trailer rear axle center, and then the coordinate vector translation from trailer coordinate to world coordinate can be expressed as:

$$P_{\mathrm{world}} = \mathrm{R}_{\mathrm{world}} * P_2 + \mathrm{T}_{\mathrm{world}} \tag{4.4}$$

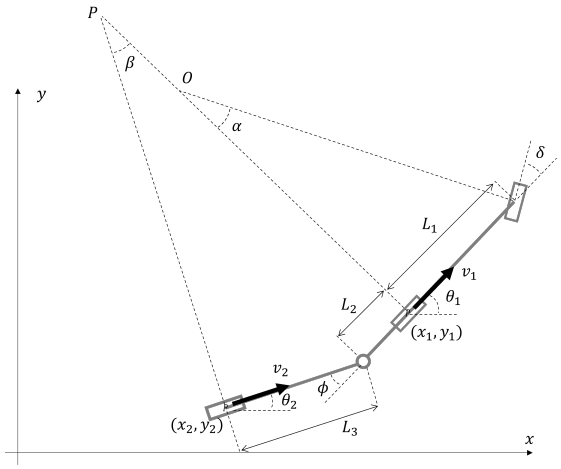


Figure 4.2: Simplified bicycle kinematic model of TTWR

In the world coordinate, the host vehicle state model is referred to as the front wheel Ackerman steering model, which allows control of the front wheel orientation relative to the heading of the vehicle. Due to limitations on the available computational power, vehicle dynamics are often highly simplified at the simulation phase [41]. Because the state model in this paper is carried out at low speed and with the assumption of an ideally flat environment, the full track model represented in Figure 4.1 can be further simplified as the kinematic bicycle model in Figure 4.2, and the host vehicle kinematic

equation can be expressed as:

$$\dot{x}_1 = v\cos\theta_1 \tag{4.5}$$

$$\dot{y}_1 = v\sin\theta_1 \tag{4.6}$$

$$\dot{\theta}_1 = \frac{v}{L_1}\tan\delta \tag{4.7}$$

The kinematic model of the trailer can be derived by propagating the state space of the host vehicle, as the trailer's kinematics are influenced by the maneuver of the host vehicle. The longitudinal speed of the trailer can be estimated by projecting the normal and longitudinal speed of the host vehicle onto the longitudinal axis of the trailer at the hitch point. Following the bicycle kinematic model, the states $[x_2, y_2, \theta_2]^T$ of the trailer can be expressed as:

$$\dot{x}_2 = v\cos\phi(1 - \frac{L_2}{L_1}\tan\phi\tan\delta)\cos\theta_2 \tag{4.8}$$

$$\dot{y}_2 = v\cos\phi(1 - \frac{L_2}{L_1}\tan\phi\tan\delta)\sin\theta_2 \tag{4.9}$$

$$\dot{\theta}_2 = -v(\frac{\sin\phi}{L_3} + \frac{L_2}{L_1 L_3}\cos\phi\tan\delta) \tag{4.10}$$

To evaluate the threat of jack-knifing, the host-trailer angle is also required for trailer state prediction, which is defined by the difference between Eqn.4.10 and Eqn.4.7:

$$\dot{\phi} = -\frac{v}{L_3}\sin\phi - \frac{v}{L_1}(1 + \frac{L_2\cos\phi}{L_3})\tan\delta \tag{4.11}$$

During reverse driving, the host-trailer system is limited to one degree of freedom in control, which is the host vehicle steering input, the system control is designed to minimize mainly the trailer's tracking error. So, the control reference variables are defined by the trailer's rear axle center position (x_2, y_2), and heading θ_2 compared with the parking slot's reference position (x_{goal}, y_{goal}), and heading angle θ_{goal}.

4.2 TTWR system physical constrains

Jack-knifing is a significant constrain in the dynamics of truck trailer wheeled robot systems. It occurs when the angle between the truck and trailer reaches a point where control over the trailer's direction is lost during reverse motion. Rather than aligning with the desired trajectory, the angle continues to grow, culminating in the trailer contacting the tractor. At this stage, the system enters an uncontrollable state, representing a critical failure mode in the control of tractor-trailer systems. Abraham [42] provides an explanation for the phenomenon of jack-knifing, identifying it as a situation where the trailer's angular velocity surpasses that of the towing vehicle, even when maximum steering input is applied. Geometrically, this is interpreted as the trailer's center of rotation moving beyond the corresponding point of the towing vehicle. The critical threshold for jack-knifing is reached when these two centers of rotation coincide, leading to a loss of control over the trailer's direction.

4.3 Controllability

Controllability is an important characteristic for the control system, which significantly impacts various control challenges, including stabilizing unstable systems through feedback and optimizing control. Complete state controllability refers to the capacity of an external input (consisting of control variables) to move the system's internal states from any starting states to any final states within a finite time frame. In simpler terms, a system is considered controllable if there exists a control input that enables the transition from any initial state to any desired final state within a specified time interval.

Consider the continuous linear system

$$\begin{aligned} \dot{x}(t) &= A(t)x(t) + B(t)u(t) \\ y(t) &= C(t)x(t) + D(t)u(t). \end{aligned} \tag{4.12}$$

where $x(t_0) = x^0$ is the initial states when the system starts.

The state pair sets (x, u) which solve Equation 4.12 for by setting the initial states (t_0, x^0) is called the behaviour $\mathcal{B}_{(A,B)}$ of Equation 4.12:

$$\mathcal{B}_{(A,B)} := \left\{ (x, u) \in X \times U \mid \exists \left(t_0, x^0\right) \in \mathbb{R} \times \mathbb{R}^n, \quad x(\cdot) = x\left(\cdot; t_0, x^0, u\right) \right\} \tag{4.13}$$

Here, X is a suitable space of functions such that

$$\left\{ x\left(\cdot; t_0, x^0, u\right) \mid \left(t_0, x^0\right) \in \mathbb{R} \times \mathbb{R}^n, \quad u \in U \right\} \subseteq X. \tag{4.14}$$

Regarding the concept of controllability, the system (A, B) is called

- Reachable at time $T > 0$ if for all $x^1 \in \mathbb{R}^n$ there exists $(x, u) \in \mathcal{B}_{(A,B)}$ such that $x(0) = 0$ and $x(T) = x^1$,
- Controllable at time T if for all $x^0, x^1 \in \mathbb{R}^n$ there exists $(x, u) \in \mathcal{B}_{(A,B)}$ such that $x(0) = x^0$ and $x(T) = x^1$,
- Null-controllable at time T if for all $x^0 \in \mathbb{R}^n$ there exists $(x, u) \in \mathcal{B}_{(A,B)}$ such that $x(0) = x^0$ and $x(T) = 0$.

To determine which states can be more or less easily controlled, it is required to examine the eigen decomposition of the Controllability Gramian. This process reveals insights into the relationships between the system states and how they respond to the

control inputs. The Controllability Gramian $W(t_0, t_1)$ is defined as:

$$W(t_0, t_1) = \int_{t_0}^{t_1} \phi(t_0, t) B(t) B(t)^T \phi(t_0, t)^T dt \tag{4.15}$$

where ϕ is the state-transition matrix.

The Equation 4.15 means if there exists a control u from state x_0 at time t_0 to state x_1 at time $t_1 > t_0$ if and only if $x_1 - \phi(t_0, t_1) x_0$ is in the column space of the Controllability Gramian.

In fact, if η_0 is a solution to $W(t_0, t_1)\eta = x_1 - \phi(t_0, t_1) x_0$ then a control given by $u(t) = -B(t)^T \phi(t_0, t)^T \eta_0$ would make the desired transfer.

Note that the matrix W defined as above has the following properties:

- $W(t_0, t_1)$ is symmetric
- $W(t_0, t_1)$ is positive semidefinite for $t_1 \geq t_0$
- $W(t_0, t_1)$ satisfies the linear matrix differential equation

$$\frac{d}{dt} W(t, t_1) = A(t) W(t, t_1) + W(t, t_1) A(t)^T - B(t) B(t)^T, W(t_1, t_1) = 0 \tag{4.16}$$

- $W(t_0, t_1)$ satisfies the equation [43]

$$W(t_0, t_1) = W(t_0, t) + \phi(t_0, t) W(t, t_1) \phi(t_0, t)^T \tag{4.17}$$

Gramians are also useful to determine the minimum-energy control $u(t)$ required to navigate the system to $x(t_f)$ at time t_f from $x(0) = 0$ [9]:

$$u(t) = B^* \left(e^{A(t_f - t)} \right)^* W_c(t_f)^{-1} x(t_f). \tag{4.18}$$

The total energy expended by this control law is given by

$$\int_0^{t_f} \|u(\tau)\|^2 d\tau = x^* W_c \left(t_f\right)^{-1} x. \tag{4.19}$$

If the controllability matrix is close to singular, more actuation enger will be required to drive the states to certian direction. On the other hand, if the eigenvalues of W_c are all large, then the system is easily controlled.

It is generally impractical to compute the Gramians directly using Equation 4.15. Instead, the Controllablity Gramian is the solution to the following Lyapunov equation:

$$AW_c + W_c A^* + BB^* = 0, \tag{4.20}$$

while the observability Gramian is the solution to

$$A^* W_o + W_o A + C^* C = 0. \tag{4.21}$$

Obtaining Gramians by solving a Lyapunov equation is typically quite expensive for high-dimensional systems. Instead of this computationally demanding approach, Gramians are frequently approximated empirically. This approximation is achieved using snapshot data gathered from both the direct system and the adjoint system, providing a more efficient means to estimate these important mathematical constructs.

In practice, the Controllability Gramian requires the integration of a system's state-transition matrix. For a simpler assessment of system controllability, a rank condition can be used that is similar to the Kalman rank condition found in time-invariant systems, which shows that the Controllability of a LTI system is determined entirely by the column space of the controllability matrix $\mathcal{C}$:

$$\mathcal{C} = \begin{bmatrix} B & AB & A^2\,B & \cdots & A^{n-1}\,B \end{bmatrix}. \tag{4.22}$$

If the matrix $\mathcal{C}$ has n linearly independent columns, meaning that it spans all of $\mathbb{R}^n$, then the system is considered controllable. The span of the columns of the controllability matrix$\mathcal{C}$ forms a Krylov subspace, which identifies the directions in $\mathbb{R}^n$ that can be manipulated with control. Consequently, controllability not only implies the possibility of arbitrary eigenvalue placement but also ensures that any state $\xi \in \mathbb{R}^n$ can be reached within a finite time using a specific actuation signal $u(t)$ [9].

In general, for a system (A, B) the following statements are equivalent:

- There exists $T > 0$ such that (A, B) is controllable at time T.
- $\text{im}(K(A, B)) = \mathbb{R}^n$
- $\text{rank}(K(A, B)) = n$
- For all $T > 0$ the system (A, B) is controllable at time T. We say that (A, B) is controllable.

4.4 TTWR system analysis

To identify the properties and conditions of the truck trailer system, its equilibrium, observability, and controllability are analyzed. Let $x = [y_2, \theta_2, \phi]^\intercal$ be the trailer states obtained from Equation 4.8 - 4.10 by neglecting the longitudinal component x_3, the state space model can be written as [44]:

$$\dot{x} = v(\mathcal{A}(x) + \mathcal{B}(x; \delta)) \tag{4.23}$$

The equilibrium state of x is the origin $x_e = 0$, and we can derive the trailer equilibrium state from Equation 4.11:

$$x_e = \{(y_2, \theta_2, \phi) \mid \delta = 0, y_2 \in \mathbb{R}, \theta_2 \in [-\pi, \pi), \phi = 0\} \tag{4.24}$$

The equilibrium set indicates during reverse driving, the trailer system is in an equilibrium state when the host vehicle and trailer are aligned in either direction, and there should be no steering angle input given by the host vehicle.

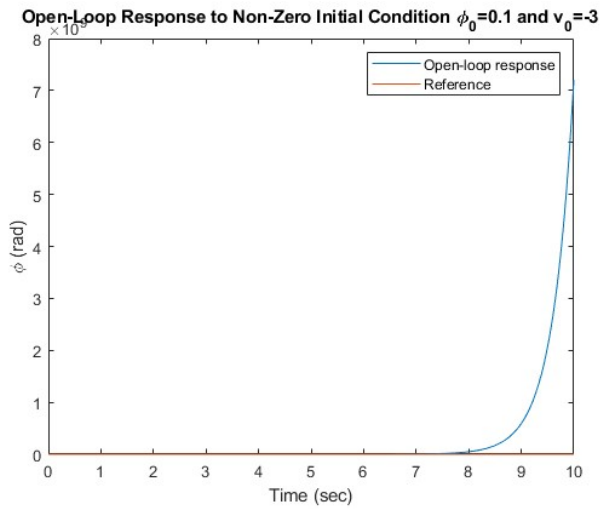
To examine the internal asymptotic stability, we substitute $\delta = 0$ into Equation 8, resulting in the following equation:

$$\dot{x} = v(\mathcal{A}x + \mathcal{B}\delta) \tag{4.25}$$

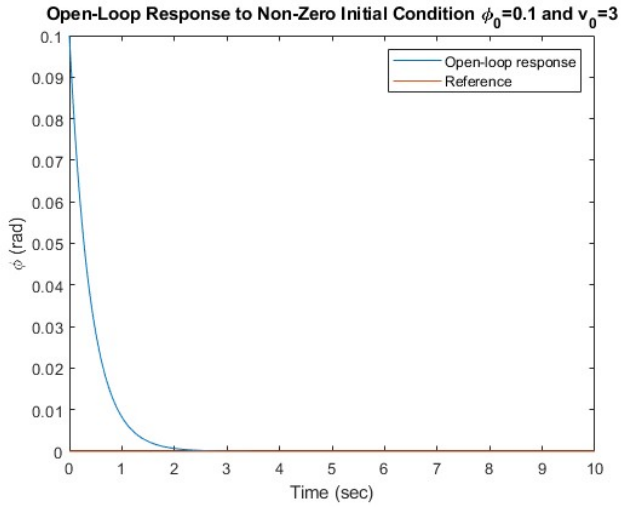
where

$$\mathcal{A} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & -\frac{1}{L_3} \\ 0 & 0 & -\frac{1}{L_3} \end{bmatrix} \mathcal{B} = \begin{bmatrix} 0 \\ \frac{L_2}{L_1 L_3} \\ -\frac{L_3 + L_2}{L_1 L_3} \end{bmatrix} \tag{4.26}$$

By using the Cayley-Hamilton theorem, the controllability matrix $\mathcal{C} = [B, AB, AAB]$ is row full rank for state space in Equation 4.25, which shows the host-trailer system to be controllable out of the singular locus [45], and the origin of the nonlinear system can be made an asymptotically stable equilibrium by state feedback.



(a) The positive velocity indicates stable system behaviour when teh TTWR can correct its Φ value



(b) The negative velocity will cause unstable system performance, the Φ angle keeps increasing until reach jacknifing

Figure 4.3: Open loop respond comparison between the positive and negative truck velocity

Chapter 5

Classical TTWR reverse parking control system

In the first stage of this research, a traditional hierarchical control system has been developed as a baseline for the future benchmark testing, by using Dubins path planning and the LQR controller. This section introduces details of this implementation for the TTWR reverse parking control. The usage of Dubins path planning ensures a minimal turning radius limitation, providing a smooth and efficient trajectory for the TTWR. Coupled with this, the LQR controller optimizes steering responses for the path following. Together, these methodologies not only streamline the reverse driving process but also enhance the precision and reliability of the control system, ensuring optimal performance in real-world scenarios.

5.1 Dubin path planning

The Dubins path, designed with a consistent velocity, operates with a singular control variable: its steering. The Dubins path is constructed by establishing common tangents between two circular arcs. These tangents can either connect the arcs from the outside (external tangents), or they can intersect the arcs diagonally, termed as internal tangents. This section will primarily focus on the Dubins path construction using an external tan-

gent. The methodology for the internal tangent follows a similar logic. Figure 5.1 showed the geometric construction of the Dubins path. The dubins path has three distinct states: maximum right turn (R), maximum left turn (L), and straight (S). Previous research from Dubins [46] has shown that optimal trajectory between two points with certain direction can be represented using six specific control sequences: RSR, RSL, RLR, LSL, LSR, and LRL. For simplification, the curvatures resulting from R and L can be labeled as C, leading to sequences like CSC and CCC.

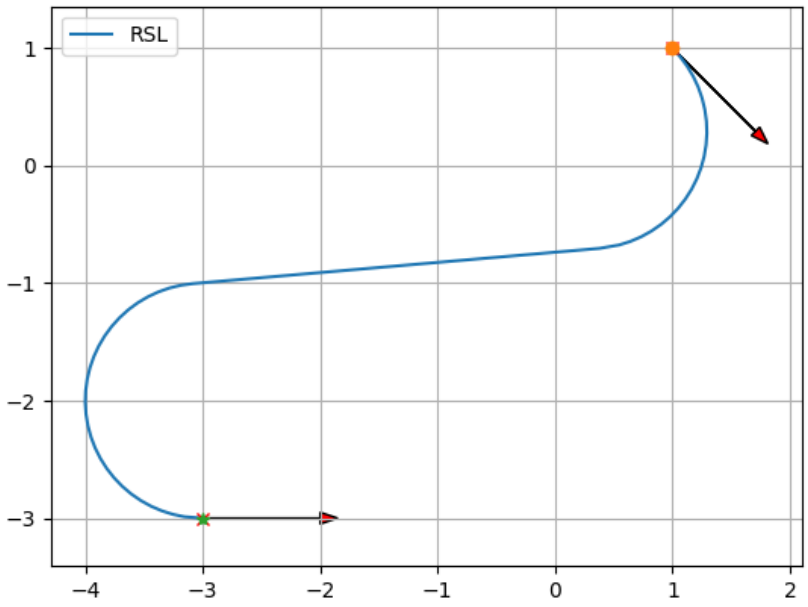
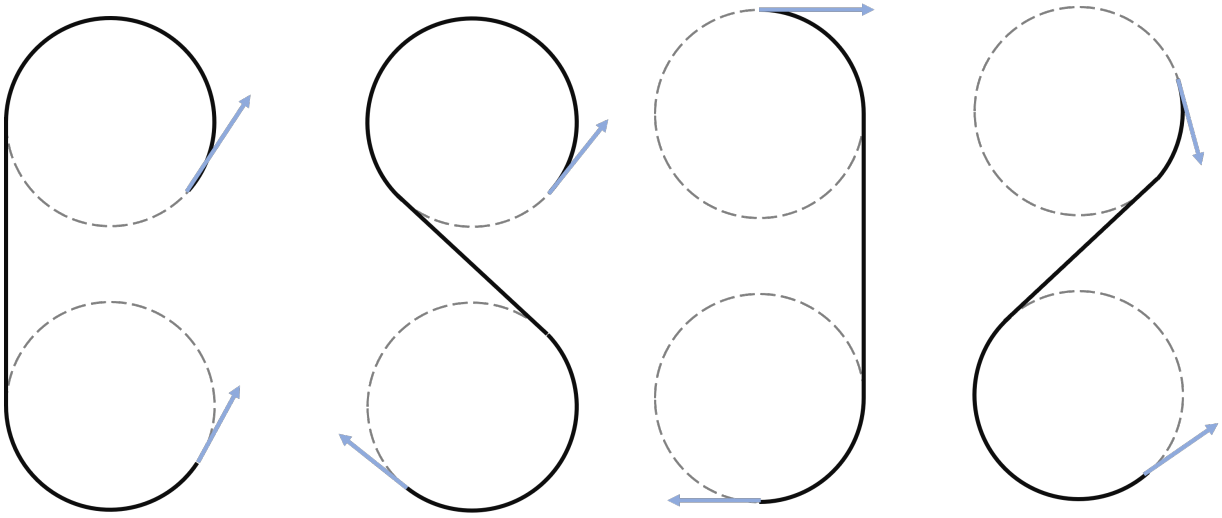


Figure 5.1: Dubins path starting from point (1, 1) with heading angle $\frac{-\pi}{4}$ and ending at (-3, -3) with heading angle 0

5.1.1 CSC Dubins Path

The CSC trajectories include RSR, LSR, RSL, and LSR, which stands for a turn followed by a straight line followed by another turn (Shown in Figure 5.2).

Pick a position and orientation for your start and goal configurations. Draw your start and goal configurations as points in the plane with arrows extending out in the direction



(a) dubins pattern LSL

(b) dubins pattern LSR

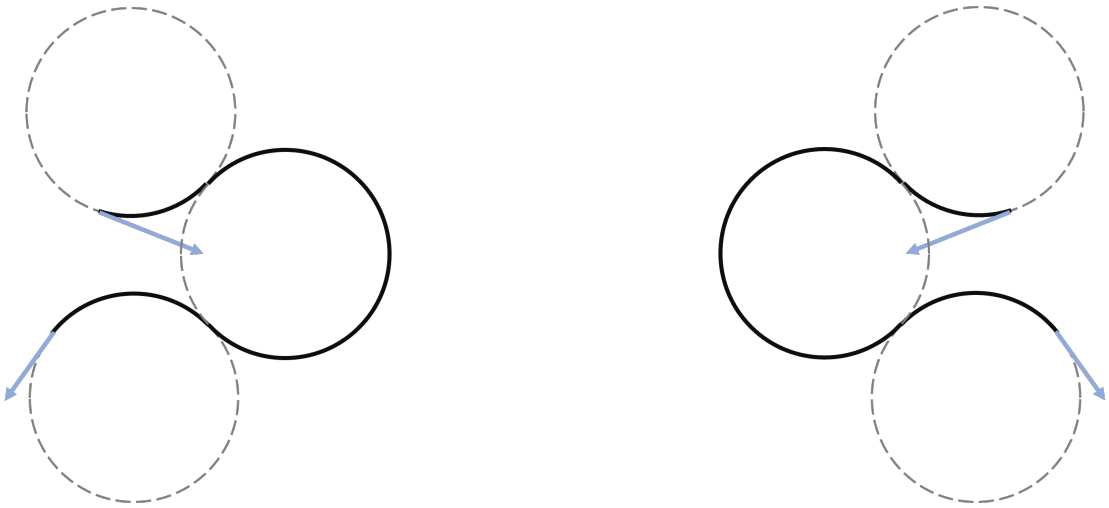
(c) dubins pattern RSR

(d) dubins pattern RSL

Figure 5.2: Four patterns under CSC type

the car is facing. Next, draw circles to the left and right of the car with radius r_{min} . The circles should be tangent at the location of the car. Draw tangent lines from the circles at the starting configuration to the circles at the goal configuration. In the next section I'll discuss how to compute this, but for now just draw them.

For every pair of circles, whether it's RR, LL, RL, or LR, four potential tangent lines can be drawn. However, only one of these lines for each pair is valid. Specifically, for the RR circles, a single line drawn from the agent's circle intersects the goal's circle in a manner that ensures the correct direction, whose Dubins trajectory can be simplified as RR without S in between. As a result, for any CSC Trajectory, there exists a distinct tangent line to be followed. This line represents the 'S' segment of the trajectory. The points where this line touches the circles are the critical points the agent needs to navigate through to complete its path. Essentially, determining these trajectories hinges on accurately identifying these tangent points.



(a) dubins pattern LRL (b) dubins pattern RLR

Figure 5.3: Four patterns under CCC type

5.1.2 CCC Dubins path

The CCC trajectories is different from CSC in the connection path, which only involve an initial turn, followed by a turn in the opposite direction, and then revert to the initial turning direction, as exemplified by the RLR Trajectory in Figure 5.3. These trajectories are applicable when the ego vehicle and the target position are close within a threshold. If they aren't, one of the circles would necessitate a radius exceeding r_{min} . In such cases, the CCC Trajectory becomes less than ideal. For the Dubin's path planning, there are only two types of CCC trajectories: RLR and LRL. Simply determining the tangent lines between RR or LL circles isn't sufficient in this context. The third circle between the starting and finishing circle in the trajectory remains tangent to both the starting and finishing turning circles. However, the points of tangent differ from those derived from standard tangent line computations.

5.1.3 Construction of Dubins path

Consider the following input parameters.

1. Initial pose of the target: $P_s\,(x_s, y_s, \theta_s)$

2. Final pose of the target: $P_f\left(x_f, y_f, \theta_f\right)$

3. Initial turning radius: $\rho_s \left(= \frac{1}{\kappa_s}\right)$

4. Final turning radius: $\rho_f \left(= \frac{1}{\kappa_f}\right)$

where the initial turning radisu and final turning radius is equal for vehicle path planning, which is determined by vehicle steering angle:

$$R = \frac{L}{tan(\delta)} \tag{5.1}$$

To compute the Dubins path, we need to start from the two turning circle center $O_s\left(x_{cs}, y_{cs}\right)$ and $O_f\left(x_{cf}, y_{cf}\right)$, which can be computed as:

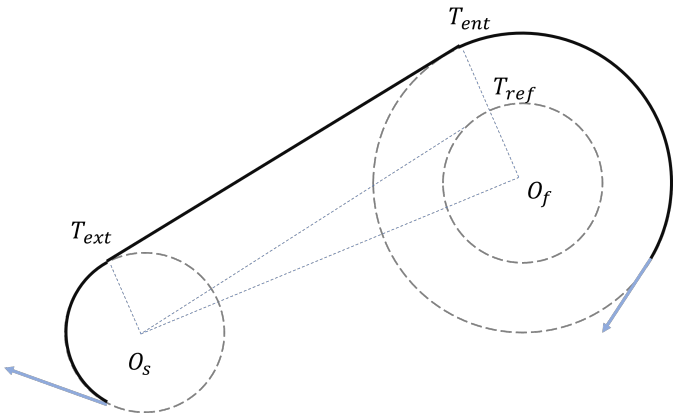
$$\begin{aligned} &(x_{cs}, y_{cs}) = \\ &(x_s \pm \rho_s \cos\left(\theta_s \pm \frac{\pi}{2}\right), y_s \pm \rho_s \sin\left(\theta_s \pm \frac{\pi}{2}\right)) \end{aligned} \tag{5.2}$$

$$\begin{aligned} &(x_{cf}, y_{cf}) = \\ &(x_f \pm \rho_f \cos\left(\theta_f \pm \frac{\pi}{2}\right), y_f \pm \rho_f \sin\left(\theta_f \pm \frac{\pi}{2}\right)) \end{aligned} \tag{5.3}$$

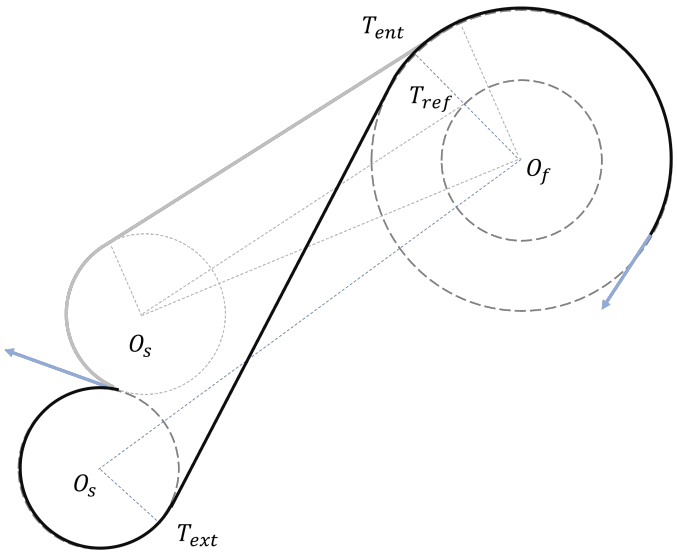
Then, the reference circle of radius $|\rho_f - \rho_s|$ is constructed at O_f for $\rho_s \leq \rho_f$, and the two primary circle center is connected with center reference line:

$$|c| = \sqrt{\left(x_{cs} - x_{cf}\right)^2 + \left(y_{cs} - y_{cf}\right)^2} \tag{5.4}$$

A perpendicular line to c at O_f intersects the reference circle at a point T' and the primary circle C_f at the tangent entry point T_{ent}. A line from O_s to T' is drawn, and another line parallel to $O_f T_{ent}$ is extended from O_s to meet C_s at the tangent exit point T_{ext}. The path is completed by connecting the starting position P_s to T_{ext} with an arc of



(a) Dubins construction following RSR pattern



(b) Dubins construction following LSR pattern, the pattern for RSR is in grey color for comparison

Figure 5.4: Different Dubins contruction pattern for the same start and finish condition

radius ρ_s, and from T_{ent} to the finishing position P_f with an arc of radius ρ_f. The resulting composite path consists of the starting arc P_sT_{ext}, the external tangent line $T_{ext}T_{ent}$, and the ending arc $T_{ent}P_f$. To determine the shortest Dubins path between the same initial and final points, multiple patterns are considered in Figure 5.4, and the one with the least total length is selected.

5.2 LQR controller design for TTWR

The first control algorithm implemented in the traditional hierarchy TTWR controller is the Linear Quadratic Regulator (LQR). As defined in Section 4.4, the system $(\mathcal{A}, \mathcal{B})$ is controllable by analysing its controllaiblity grammian or by applying Cayley-Hamilton theorem to its controllability matrix, which indicates that it is possible to manipulate the eigen values of the closes loop system $(\mathcal{A} - \mathcal{B}K)$ by choosing the state feed back control law $u = -Kx$. Similarly, the LQR algorithm is a widely used algorithm to find an appropriate state-feedback controller. LQR controllers possess inherent robustness with guaranteed gain and phase margin [47], and LQR plays a crucial role in addressing the LQG (linear–quadratic–Gaussian) challenge, which is the foundational problems in control theory.

For a controllable system, given either full-state measurements or full-state estimate, there exists more than one control laws, $u = -Kx$, which stabilize the system states, and the eigenvalues of the closed-loop system, $(A - BK)$, can be designed and placed to left-half of the complex plane to achieve system stability. However, over stability by choosing large negative eigenvalues might require significant large control efforts, and may exceed the system input limitation, cause a wast of control energy, and reduce the system robustness for certain noise and disturbance. Therefore, the balance between stability of the closed loop system and the aggressiveness of the control input is the main purpose in optimal control, which is achieved by choosing the most suitable gain matrix K to stabilize the system without expending too much control effort. Three rules shall be taken into consideration when choosing the proper control gain, which includles [9]:

- prevent the controller from overreacting to high-frequency noise and disturbances
- the control actuation does not exceed limitation
- the control effort is not prohibitively expensive

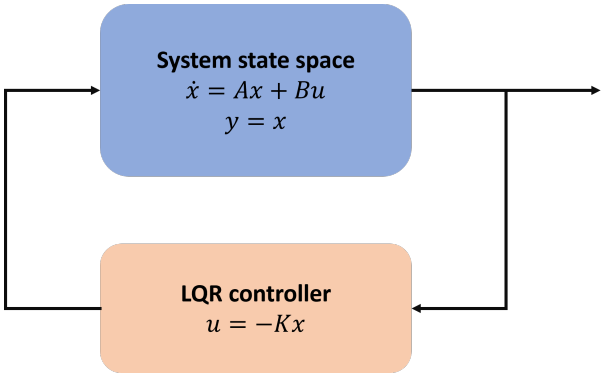


Figure 5.5: LQR controller schematic diagram

For an infinite-horizon, continuous-time system described in Equation 4.12, the cost function defined as:

$$J = \int_0^\infty \left(x^T Q x + u^T R u + 2x^T N u\right) dt \tag{5.5}$$

where the matrices Q and R weight the cost of state deviation and actuation cost respectively, the matrix Q is positive semi-definite, and R is positive definite; these matrices are often diagonal, and the diagonal elements can be tuned to change the relative importance of the control objectives.

The weight matrix is useful to make choices to optimize the control law, for example, a larger value in a particular diagonal entry of Q matrix means that deviations in the corresponding state will be penalized more heavily in the cost function. This will lead the controller to work harder to regulate the particular state. When the Q matrix is set to be zero, the controller will not attempt to regulate the state at all, focusing solely on minimizing control effort. The matrix R determines how much the control actions are penalized in the cost function. A larger value in a diagonal entry of R means that the corresponding control action will be penalized more. This discourages aggressive control actions and can be used to ensure smoother control signals. If R is too small, the controller might become overly aggressive, leading to large control signals that could saturate actuators or cause wear and tear.

As shown in Figure 5.5, the LQR control law $u = -Kx$ is designed to minimize

the cost function $J = \lim_{t\to\infty} J(t)$. As its name suggests, LQR is a control strategy formulated for linear systems and employs the linear control law to minimize a cost function defined quadratically, aims to optimally regulate the system's state $\lim_{t\to\infty} \mathbf{x}(t) = 0$. The analytical solution for the optimal controller gain K can be computed following:

$$K = R^{-1}\left(B^T P + N^T\right) \tag{5.6}$$

and the variable P is found by solving the continuous time algebraic Riccati equation:

$$A^T P + PA - (PB + N)R^{-1}\left(B^T P + N^T\right) + Q = 0 \tag{5.7}$$

which can be also written as:

$$\mathcal{A}^T P + P\mathcal{A} - PBR^{-1}B^T P + \mathcal{Q} = 0 \tag{5.8}$$

with

$$\mathcal{A} = A - BR^{-1}N^T \quad \mathcal{Q} = Q - NR^{-1}N^T \tag{5.9}$$

For the TTWR control topic, the state equation can be formulated as:

$$\begin{bmatrix} \Delta\dot{\theta}_1 \\ \Delta\dot{\theta}_2 \\ \Delta\dot{y} \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 \\ \frac{v}{L_3} & -\frac{v}{L_3} & 0 \\ 0 & V & 0 \end{bmatrix} \begin{bmatrix} \Delta\theta_1 \\ \Delta\theta_2 \\ \Delta y \end{bmatrix} + \begin{bmatrix} \frac{v}{L_1} \\ -\frac{L_2}{L_1 L_3} v \\ 0 \end{bmatrix} \delta \tag{5.10}$$

as shown in Figure 5.6, the reference states error consist of $\Delta\theta_1$, $\Delta\theta_2$, and Δy, which represents the heading error between TTWR's truck heading angle and the reference point heading angle, the heading error between trailer heading angle and reference point heading angle, and the lateral distance in trailer coordinate between trailer position and refernece point.

To compute the optimal gain of the TTWR controller, MATLAB provides built-in

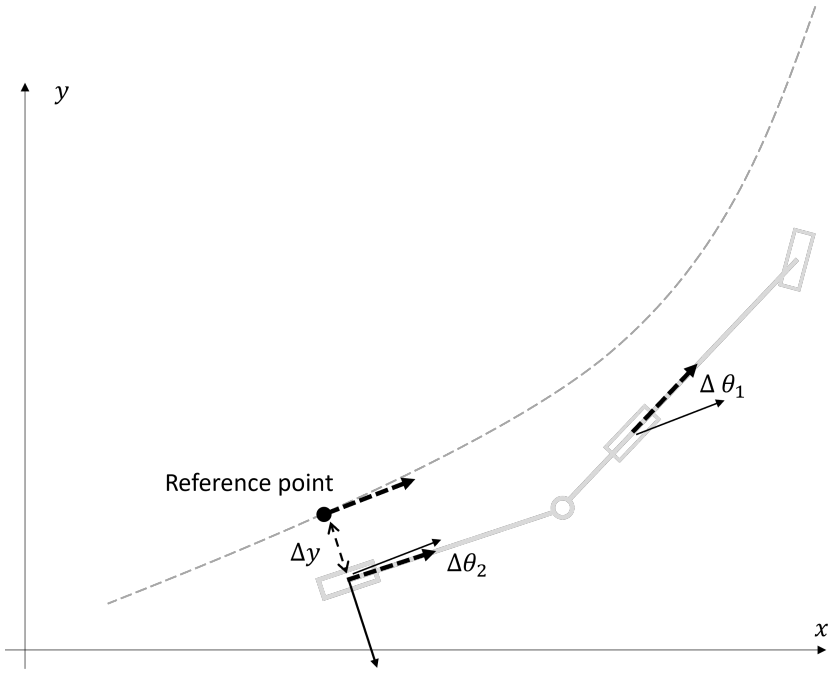


Figure 5.6: LQR controller reference states error

functions which can solve Riccati equation efficiently by providing the system state space model, weight matrix Q and R, the code is as following:

```
% TTWR parameters
L1 = 3; % truck wheelbase length [m]
L2 = 2; % hitch length [m]
L3 = 1.5; % trailer wheelbase length [m]
vx = -2; % truck longitudinal velocity [m/s]

% Linearized State Space
A = [0        0          0;
     vx./L3   -vx./L3    0;
     0        vx         0];

B = [vx./L1;
     -L2*vx ./ (L1*L3);
```

```
      0];

C = eye(3);
D = zeros(3, 1);

% state states x = [\theta1, \theta2, \y_err]
sys = ss(A, B, C, D);

% State weight matrix
% \theta1 and \theta2 error ref for 2 degrees, and
% lateral error reference value is 0.8
Q = [1/(deg2rad(2).^2)        0                          0;
     0                    1/(deg2rad(2).^2)             0;
     0                         0                  1/(0.8.^2)
        ];
% Input weight matrix for steering angle
R = 1 / (deg2rad(steer_max).^2);

% Output K is the LQR gain
[K,S,P] = lqr(sys, Q, R)
```

The results of the LQR TTWR reverse path following control is shown in Figures 5.7, which depict the reference trajectory and the TTWR vehicle states respectively. The blue line is the Dubins trajectory started with initial states $(25, 25, -\frac{\pi}{3})$ and final states $(-25. - 10, 0)$ which is lateral offset between trajectory and truck or trailer, and the heading angle difference between trajectory and trailer. The Figure 5.8 shows the TTWR states along the path following task, and the lateral offset of truck and traler are both

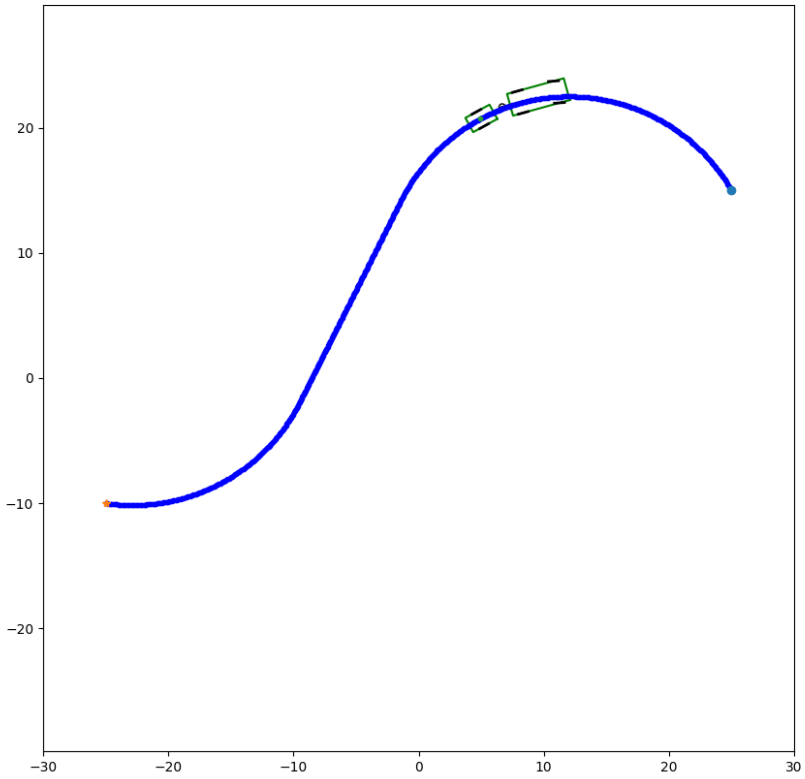


Figure 5.7: The simulation of using Dubins and LQR for TTWR reverse driving control

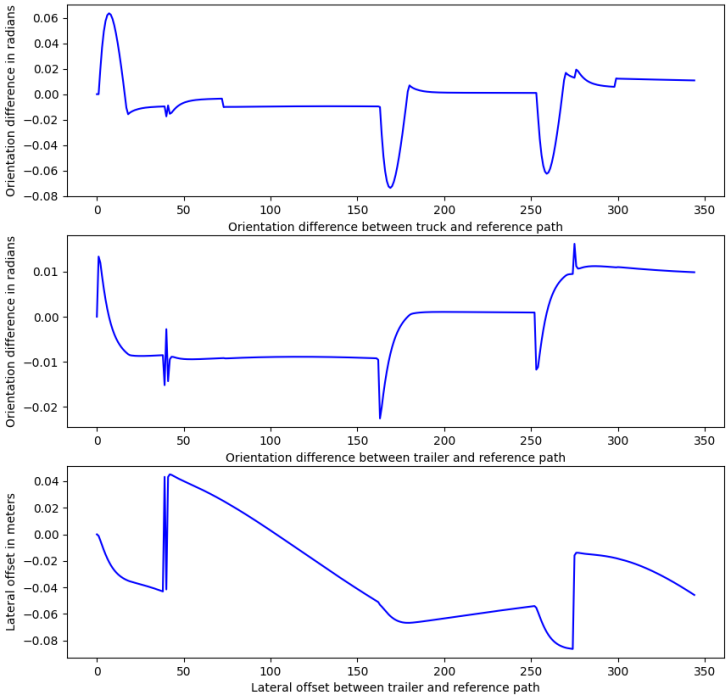


Figure 5.8: The system states of the LQR controller

within 0.1 meters, and the heading angle difference is also within 0.1 rad. The LQR controller effectively ensures that the articulated vehicle system follows closely to the desired trajectory. This is evident from the gentle turning motions observed whenever the blue line alters its direction. Moreover, the consistent spacing between each sample signifies a gradual and smooth turning process, highlighting the controller's efficiency in maintaining steady and controlled maneuvers.

5.3 MPC controller design for TTWR

Model Predictive Control (MPC) operates by executing finite-horizon optimization for a system model at each discrete time step. The primary idea behind MPC is to use a model of the system to predict its future behavior over a finite horizon. Based on this prediction, the control inputs are determined by minimizing a cost function, which typically represents a combination of tracking errors and control efforts.

$$J = \sum_{k=0}^{N-1} \left(x(k)^T Q x(k) + u(k)^T R u(k) \right) + x(N)^T P x(N) \tag{5.11}$$

where J is the cost to be minimized, N is the prediction horizon, X_k is the system states at time t_k, U_k is the input states at time t_k, and Q and R is the weight matrix for system states and control input respectively, and these two matrices are designed to be positive definite, and the P is the terminal cost wight matrix.

At every instance t_k, the system is sampled, and a control strategy is derived for the control horizon $[t_k, t_k + N_C]$ to minimize the deviation from the desired trajectory over the prediction horizon $[t_k, t_k + N_P]$. Beyond N_C control moves, the control input remains constant for the rest of the prediction horizon. The system's progression is forecasted using an intrinsic mathematical model, and a numerical optimization technique minimizes the control cost. Only the initial control input from the determined sequence is implemented at each time step t_k. This procedure is repeated in every cycle, with both the control and prediction horizons advancing forward. The principle of the MPC algorithm is shown in Figure 5.9.

To describe the TTWR system state space trasfer function, a new state variable v_1 is added for referencing, and then the state vector becomes $X = [x_1, y_1, \theta_1, x_2, y_2, theta_2, phi, v]$, and the input states change from $U = [v, \delta]$ to $U = [a, \delta]$ where a is the acceleration of the truck vehicle. The new state representation is as following:

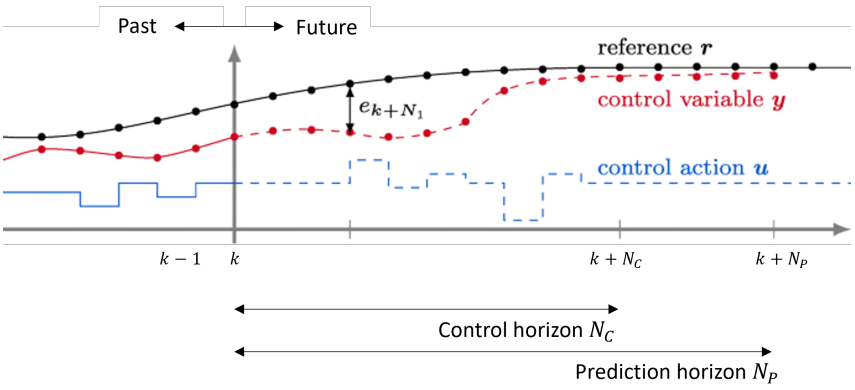


Figure 5.9: Function principle of a model-based predictive [48]

$$
\begin{aligned}
\dot{x}_1 &= f_1(X,U) = v\cos\theta_1 \\
\dot{y}_1 &= f_2(X,U) = v\sin\theta_1 \\
\dot{\theta}_1 &= f_3(X,U) = \frac{v}{L_1}\tan\delta \\
\dot{x}_2 &= f_4(X,U) = v\cos\phi(1-\frac{L_2}{L_1}\tan\phi\tan\delta)\cos\theta_2 \\
\dot{y}_2 &= f_5(X,U) = v\cos\phi(1-\frac{L_2}{L_1}\tan\phi\tan\delta)\sin\theta_2 \\
\dot{\theta}_2 &= f_6(X,U) = -v(\frac{\sin\phi}{L_3}+\frac{L_2}{L_1L_3}\cos\phi\tan\delta) \\
\dot{\phi} &= f_7(X,U) = -\frac{v}{L_3}\sin\phi-\frac{v}{L_1}(1+\frac{L_2\cos\phi}{L_3})\tan\delta \\
\dot{v_{11}} &= f_8(X,U) = a
\end{aligned}
\tag{5.12}
$$

To get the Jacobian matrix A' of the TTWR system, compute the partial derivatives of state transfer function f_1 to f_8 with respect to each elements in the state vector X:

$$A' = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial y_1} & \frac{\partial f_1}{\partial \theta_1} & \frac{\partial f_1}{\partial x_2} & \frac{\partial f_1}{\partial y_2} & \frac{\partial f_1}{\partial \theta_2} & \frac{\partial f_1}{\partial \phi} & \frac{\partial f_1}{\partial v_1} \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial y_1} & \frac{\partial f_2}{\partial \theta_1} & \frac{\partial f_2}{\partial x_2} & \frac{\partial f_2}{\partial y_2} & \frac{\partial f_2}{\partial \theta_2} & \frac{\partial f_2}{\partial \phi} & \frac{\partial f_2}{\partial v_1} \\ \frac{\partial f_3}{\partial x_1} & \frac{\partial f_3}{\partial y_1} & \frac{\partial f_3}{\partial \theta_1} & \frac{\partial f_3}{\partial x_2} & \frac{\partial f_3}{\partial y_2} & \frac{\partial f_3}{\partial \theta_2} & \frac{\partial f_3}{\partial \phi} & \frac{\partial f_3}{\partial v_1} \\ \frac{\partial f_4}{\partial x_1} & \frac{\partial f_4}{\partial y_1} & \frac{\partial f_4}{\partial \theta_1} & \frac{\partial f_4}{\partial x_2} & \frac{\partial f_4}{\partial y_2} & \frac{\partial f_4}{\partial \theta_2} & \frac{\partial f_4}{\partial \phi} & \frac{\partial f_4}{\partial v_1} \\ \frac{\partial f_5}{\partial x_1} & \frac{\partial f_5}{\partial y_1} & \frac{\partial f_5}{\partial \theta_1} & \frac{\partial f_5}{\partial x_2} & \frac{\partial f_5}{\partial y_2} & \frac{\partial f_5}{\partial \theta_2} & \frac{\partial f_5}{\partial \phi} & \frac{\partial f_5}{\partial v_1} \\ \frac{\partial f_6}{\partial x_1} & \frac{\partial f_6}{\partial y_1} & \frac{\partial f_6}{\partial \theta_1} & \frac{\partial f_6}{\partial x_2} & \frac{\partial f_6}{\partial y_2} & \frac{\partial f_6}{\partial \theta_2} & \frac{\partial f_6}{\partial \phi} & \frac{\partial f_6}{\partial v_1} \\ \frac{\partial f_7}{\partial x_1} & \frac{\partial f_7}{\partial y_1} & \frac{\partial f_7}{\partial \theta_1} & \frac{\partial f_7}{\partial x_2} & \frac{\partial f_7}{\partial y_2} & \frac{\partial f_7}{\partial \theta_2} & \frac{\partial f_7}{\partial \phi} & \frac{\partial f_7}{\partial v_1} \\ \frac{\partial f_8}{\partial x_1} & \frac{\partial f_8}{\partial y_1} & \frac{\partial f_8}{\partial \theta_1} & \frac{\partial f_8}{\partial x_2} & \frac{\partial f_8}{\partial y_2} & \frac{\partial f_8}{\partial \theta_2} & \frac{\partial f_8}{\partial \phi} & \frac{\partial f_8}{\partial v_1} \end{bmatrix} \tag{5.13}$$

The result of the partial derivatives is computed below, where the partial derivatives not listed are equal to zero:

$$
\begin{aligned}
\frac{\partial f_1}{\partial \theta_1} &= -v_1 \sin\theta_1 \\
\frac{\partial f_1}{\partial v_1} &= cos\theta_1 \\
\frac{\partial f_2}{\partial \theta_1} &= v_1 sin\theta_1 \\
\frac{\partial f_2}{\partial v_1} &= sin\theta_1 \\
\frac{\partial f_3}{\partial v_1} &= \frac{\tan\delta}{L_1} \\
\frac{\partial f_4}{\partial \theta_2} &= v\cos\phi(1 - \frac{L_2}{L_1}\tan\delta\tan\phi)(-\sin\theta_2) \\
\frac{\partial f_4}{\partial \phi} &= v(-\sin\phi)(1 - \frac{L_2}{L_1}\tan\delta\tan phi)\cos\theta_2 \\
&\quad + v\cos\phi(-\frac{L_2}{L_1} * \sec\phi\tan\delta)\cos theta_2 \\
\frac{\partial f_4}{\partial v_1} &= \cos\phi(1 - \frac{L_2}{L_1}\tan phi\tan\delta)\cos\theta_2 \\
\frac{\partial f_5}{\partial \theta_2} &= v\cos\phi(1 - \frac{L_2}{L_1}\tan\delta\tan\phi)\cos\theta_2 \\
\frac{\partial f_5}{\partial \phi} &= v(-\sin\phi)(1 - \frac{L_2}{L_1}\tan\delta\tan phi)\sin\theta_2 \\
&\quad + v\cos\phi(-\frac{L_2}{L_1} * \sec\phi\tan\delta)\sin theta_2 \\
\frac{\partial f_5}{\partial v_1} &= \cos\phi(1 - \frac{L_2}{L_1}\tan phi\tan\delta)\sin\theta_2 \\
\frac{\partial f_6}{\partial \phi} &= -v_1(\frac{\cos\phi)}{L_3} - \frac{L_2}{L_1 L_3}\sin phi\tan delta) \\
\frac{\partial f_6}{\partial v_1} &= -(\frac{\sin\phi}{L_3} + \frac{L_2}{L_1 L_3}\cos\phi\tan\delta) \\
\frac{\partial f_7}{\partial \phi} &= -\frac{v}{L_3}\cos\phi - \frac{v}{L_1}(1 - \frac{L_2\sin\phi}{L_3})\tan\delta \\
\frac{\partial f_7}{\partial v_1} &= -\frac{1}{L_3}\sin\phi - \frac{1}{L_1}(1 + \frac{L_2\cos\phi}{L_3})\tan\delta
\end{aligned}
\tag{5.14}
$$

And similarly, the matrix B' can be wrriten as:

$$B' = \begin{bmatrix} \frac{\partial f_1}{\partial \delta} & \frac{\partial f_1}{\partial a} \\ \frac{\partial f_2}{\partial \delta} & \frac{\partial f_2}{\partial a} \\ \frac{\partial f_3}{\partial \delta} & \frac{\partial f_3}{\partial a} \\ \frac{\partial f_4}{\partial \delta} & \frac{\partial f_4}{\partial a} \\ \frac{\partial f_5}{\partial \delta} & \frac{\partial f_5}{\partial a} \\ \frac{\partial f_6}{\partial \delta} & \frac{\partial f_6}{\partial a} \\ \frac{\partial f_7}{\partial \delta} & \frac{\partial f_7}{\partial a} \\ \frac{\partial f_8}{\partial \delta} & \frac{\partial f_8}{\partial a} \end{bmatrix} \tag{5.15}$$

where the partial derivatives of equations f_1 to f_8 with respect to input variables can be written as the following (similarly, the rest of the partial derivatives is 0):

$$\begin{aligned} \frac{\partial f_8}{\partial a} &= 1 \\ \frac{\partial f_3}{\partial \delta} &= \frac{v_1}{L_1} \sec \delta \\ \frac{\partial f_4}{\partial \delta} &= -v_1 \cos \phi \frac{L_2}{L_1} \tan phi \sec \delta \cos \delta \\ \frac{\partial f_5}{\partial \delta} &= -v_1 \cos \phi \frac{L_2}{L_1} \tan phi \sec \delta \sin \delta \\ \frac{\partial f_6}{\partial \delta} &= -v_1 \frac{L_2}{L_1 L_3} \cos \phi \sec \delta \\ \frac{\partial f_7}{\partial \delta} &= -\frac{v_1}{L_1}(1 + \frac{L_2}{L_3} \cos \phi) \sec \delta \end{aligned} \tag{5.16}$$

Then, using the methods of Forward Euler Discretization with sampling time dt, the state transfer equation can be written as:

$$X_{k+1} = X_k + f\left(X_k, U_k\right) dt \tag{5.17}$$

Using first degree Tayer expantion around $\bar{X}$ and $\bar{U}$

$$\begin{aligned} X_{k+1} &= X_k + \left(f(\bar{X}, \bar{U}) + A' z_k + B' U_k - A' \bar{X} - B' \bar{U}\right) dt \\ X_{k+1} &= \left(I + dt A'\right) X_k + \left(dt B'\right) U_k + \left(f(\bar{X}, \bar{U}) - A' \bar{X} - B' \bar{U}\right) dt \end{aligned} \tag{5.18}$$

The equation 5.18 can be transferd to the form following state space control schema:

$$x_{k+1} = Ax_k + BU_k + C \tag{5.19}$$

where the A, B, and C can be replaced using the derived Jacobian matrix:

$$\begin{aligned} A &= (I + dtA') \\ B &= dtB' \\ C &= (f(\bar{z}, \bar{u}) - A'\bar{z} - B'\bar{u})\, dt \end{aligned} \tag{5.20}$$

5.4 Proximal Policy Optimization based end-to-end control algorithm

* this section is from the first published journal paper [49]

The Proximal Policy Optimization (PPO) method is an improved policy gradient (PG) reinforcement learning technique introduced in 2017 by OpenAI [50]. It is characterized as an unconstrained optimization problem comprising two components: cumulative discounted return and Kullback–Leibler (KL) divergence [6]. Compared with other DRL algorithms, PPO demonstrates improved data efficiency, reduced computational demands, and a reduced likelihood of policy divergence or catastrophic memory loss.

The foundation of the PG method lies in the calculation of the policy gradient estimator and its application within a stochastic gradient ascent framework. A frequently used gradient estimator can be written as:

$$\hat{g} = \hat{\mathbb{E}}_t \left[\nabla_\theta \log \pi_\theta \left(a_t \mid s_t \right) \hat{A}_t \right] \tag{5.21}$$

where π_θ is a stochastic policy and $\hat{A}_t$ is an estimator at timestep t, and $\hat{\mathbb{E}}_t[\ldots]$ is empirical average over a finite batch of samples. The estimator $\hat{g}$ can be computed by differentiating

the objective

$$L^{PG}(\theta) = \hat{\mathbb{E}}_t \left[\log \pi_\theta \left(a_t \mid s_t \right) \hat{A}_t \right] \tag{5.22}$$

where $\hat{A}_t$ is the estimator of the advantage function at timestep t, and π_θ is a stochastic policy.

A challenge with the PG method is its sensitivity to step size, complicating the task of selecting an appropriate step size. To address this unconstrained optimization challenge, drawing inspiration from the Trust Region Policy Optimization (TRPO) method [51], the objective function is maximized using the ratio of prior strategies $\pi_{\theta_{\text{old}}}$ to new strategies π_θ, while penalizing the magnitude of the policy update with coefficient β:

$$\underset{\theta}{\text{maximize}} \hat{\mathbb{E}}_t \left[\frac{\pi_\theta \left(a_t \mid s_t \right)}{\pi_{\theta_{\text{old}}} \left(a_t \mid s_t \right)} \hat{A}_t - \beta \text{KL} \left[\pi_{\theta_{\text{old}}} \left(\cdot \mid s_t \right), \pi_\theta \left(\cdot \mid s_t \right) \right] \right] \tag{5.23}$$

where the probability ratio of old and new strategies is denoted as $r_t(\theta)$ in following equations.

To penalize the excessively large policy update caused when maximizing the objective L^{CPI}, the PPO algorithm penalizes changes to the policy that move $r_t(\theta)$ away from 1, and get the main objective function:

$$L^{CLIP}(\theta) = \hat{\mathbb{E}}_t \left[\min \left(r_t(\theta) \hat{A}_t, \text{clip} \left(r_t(\theta), 1 - \epsilon, 1 + \epsilon \right) \hat{A}_t \right) \right] \tag{5.24}$$

where the first term L^{CPI} is derived from the TRPO method, and the second term, $\text{clip} \left(r_t(\theta), 1 - \epsilon, 1 + \epsilon \right) \hat{A}_l$, represents the clip function which constrains the values of the old and new policy parameters $r_t(\theta)$ within the range $[1 - \epsilon, 1 + \epsilon]$. The final step involves the application of the min function to select the lesser value between the clipped and unclipped objectives.

The core philosophy of the PPO algorithm is to mitigate the challenges caused by large policy updates in the PG algorithm, which can it difficult for step sizes determina-

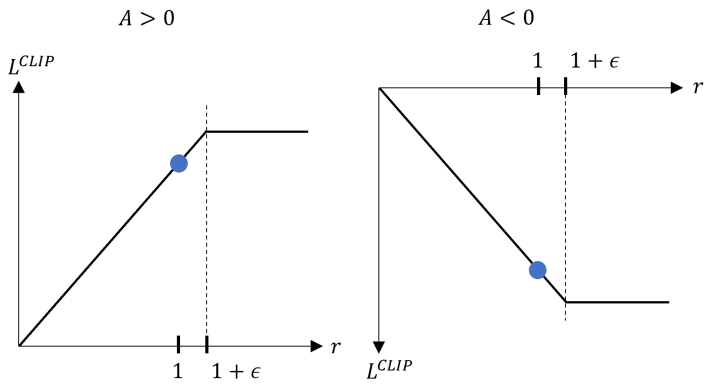


Figure 5.10: Illustration of the clip function, where the probability ratio r is clipped at $1-\epsilon$ or $1+\epsilon$ depending on the polarity of the advantage

tion and low data efficiency. By adopting a clipping mechanism to avoid imposing hard constraint completely, the PPO is proven to be very effective in dealing with a wide range of challenging tasks, while being simple to implement and tune.

The architecture for the TTWR autonomous parking mechanism is depicted in Figure 5.11. This system is bifurcated into two primary components: a profound reinforcement learning module and a simulation framework. Specifically, the profound reinforcement learning module utilizes an LSTM-PPO technique to train the agent for mastering autonomous parking and evading obstacles, all while engaging with the entities within the parking milieu. The simulation framework encompasses the kinematic model of TTWR, static vehicular obstructions, parking spaces, and sonar-driven distance sensing, all realized in Matlab to engage with the training agent.

The TTWR mechanism evaluates the parking target coordinates, TTWR system configurations, and the perceived obstacle data to execute the autonomous parking operation. To address the dimensionality challenges stemming from intricate observation data, the technique discretizes the obstacle perception data around the agent, sampling points within a predefined threshold. This allows for a more streamlined observation vector, represented as:

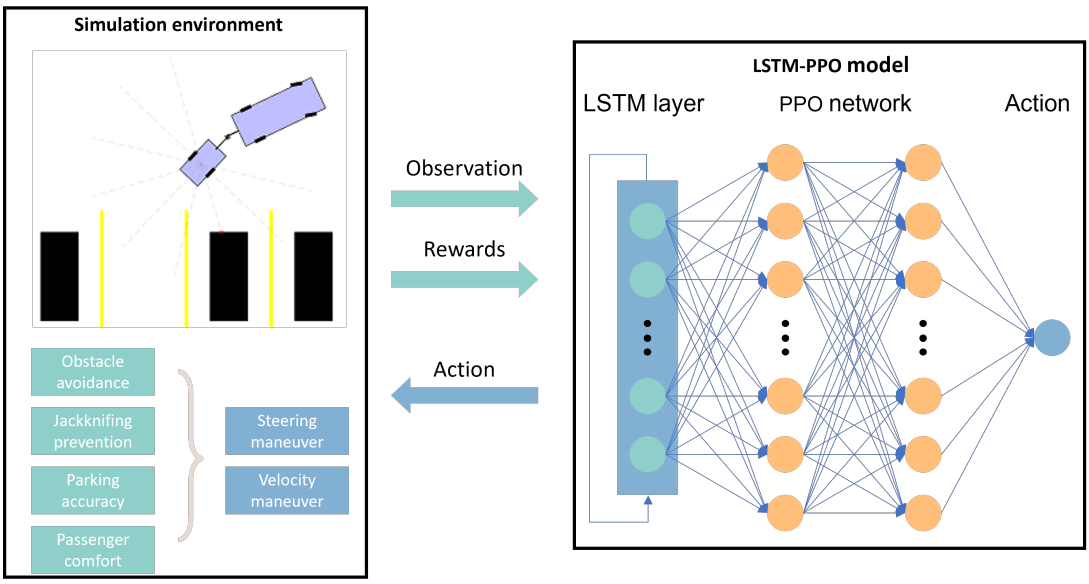


Figure 5.11: TTWR autonomous parking system layout

$$S_t = \left[\; S_{\text{TTWR}}, S_{\text{sonar}}, S_{\text{goal}} \; \right]^T \tag{5.25}$$

Here, $S_{\text{TTWR}} = \left[\; x_1, y_1, \theta_1, x_2, y_2, \theta_2, \phi \; \right]^T$ represents the TTWR state configurations as defined by the kinematic model. The parking objective, $S_{\text{goal}} = \left[\; x_{\text{goal}}, y_{\text{goal}}, \theta_{\text{goal}} \; \right]^T$, signifies the desired parking location for the TTWR, determined by user input.

Furthermore, S_{sonar} is the data from sonar instruments affixed to the trailer. Instead of a full 360° environmental scan, the sonar provides distance measurements within its field of view. These readings can be transformed from polar to Cartesian coordinates to depict obstructions around the trailer. Specifically, the sonar readings, S_{sonar}, can be articulated as:

$$S_{\text{sonar}} = \left[\; dist_1 \quad dist_2 \quad \cdots \quad dist_n \; \right] \tag{5.26}$$

To enhance the driving experience and ensure fluid vehicle control, this study introduces an action module that represents the primary vehicle's steering input for lateral management. The steering controller prompts the agent to select the optimal action

value from a continuous action domain, produced by the LSTM-PPO algorithm, and is defined within the range of $[-\delta_{max}, \delta_{max}]$, where δ_{max} is the utmost steering angle based on the primary vehicle's specifications.

The reward function acts as the feedback conduit for the controller, associating the current state and action of the TTWR to a scalar metric that evaluates potential maneuvers. Specifically, the reward function is crafted to guide the controller in learning how to adeptly park the trailer, emphasizing collision prevention and ensuring a seamless user experience.

- Distance metric R_{dist}: Assesses the gap between the trailer's position and the parking space;
- Parking metric R_{parking}: Rewards the trailer when parked within the slot, adhering to specified tolerances;
- Steering constraint R_{δ}: Penalizes excessive steering maneuvers to bolster passenger comfort;
- Collision constraint $R_{\text{collision}}$: Penalizes when the TTWR system nears surrounding entities;
- Jackknife avoidance R_{ϕ}: Assesses the risk when nearing a jackknifing scenario.

As illustrated in Figure 5.11, the distance metric R_d is provided in every iteration to motivate the TTWR to gravitate towards the designated parking slot. This is defined as:

$$R_{\text{dist}}(s_t, a_t, s_{t+1}) = e^{-(s_t^2 + s_{\text{goal}}^2)} \tag{5.27}$$

The parking metric R_{park} is allocated when the trailer adeptly parks itself within the parking slot, adhering to state tolerance s_{ϵ}. The additional accuracy metric is gauged by the distance between the concluding state s_{final} and goal state s_{goal}:

$$R_{\text{parking}} = \begin{cases} c - \frac{\|s_{\text{final}} - s_{\text{goal}}\|}{s_\epsilon}, \text{ if } \|s_{\text{final}} - s_{\text{goal}}\| \leq s_\epsilon \\ 0, \text{ otherwise} \end{cases} \tag{5.28}$$

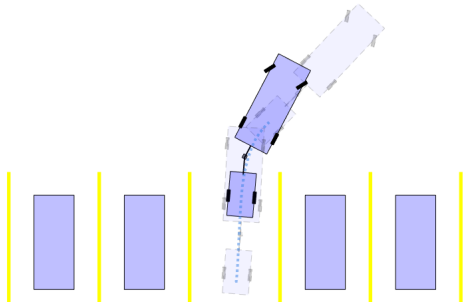
Additionally, the collision penalty R_c is applied when the distance between TTWR and surrounding obstructions falls within a threshold, ending the ongoing episode to indicate a collision. The jackknife penalty R_ϕ is proportional to the ϕ value, promoting minor maneuvers, and the steering penalty is proportional to the steering value to deter large δ inputs.

The comprehensive parking reward, encompassing all four components, is expressed in Eqn. 5.29:

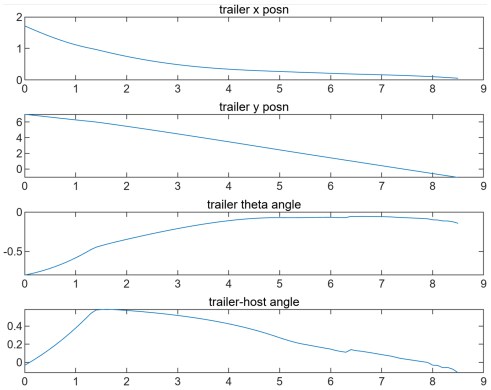
$$R = R_{\text{collision}} + R_{\text{parking}} + R_{\text{collision}} + R_\delta + R_\phi \tag{5.29}$$

Figure 5.12 showcases the trajectory of perpendicular parking simulation outcomes, validating that the proposed LSTM-PPO controller can adeptly modify the TTWR system's inputs, including host velocity and steering angle, to approach the chosen parking spot during the perpendicular parking task. The starting positions are initialized with a certain degree of noise, and the goal position is determined by user input. The dotted trajectory represents the trailer's path, with two faded TTWRs indicating its initial and concluding positions. In Figure 5.12b, the trailer's states during the parking task are displayed in the global coordinate system, aligning with the chosen parking spot's coordinates and orientation.

In Figure 5.13, the TTWR trajectory during a 45° oblique parking simulation is depicted. Similarly, the starting positions are initialized with a certain degree of noise, and the proposed methods successfully parked the TTWR system, ensuring jackknife prevention and obstacle evasion.

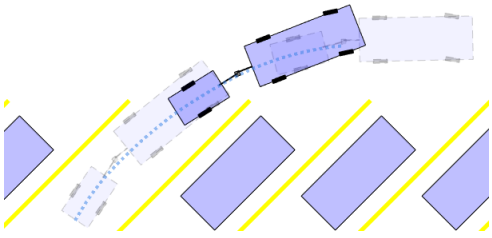


(a) Perpendicular parking simulation

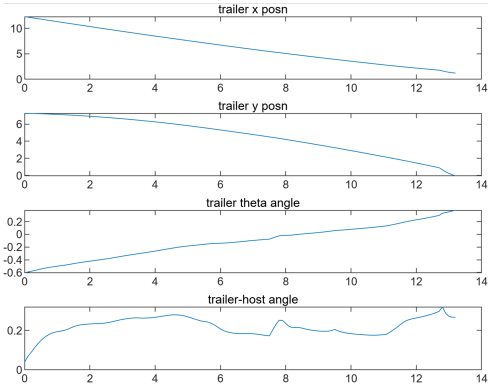


(b) Perpendicular parking system states

Figure 5.12: TTWR perpendicular parking



(a) Oblique parking system states



(b) Oblique parking system states

Figure 5.13: TTWR oblique parking

5.5 Trajectory state model-based reinforcement learning for TTWR reverse parking

* this section is from the submitted second paper which does not provide citation for now.

By employing the trajectory state prediction model as the foundation model to interact with the Proximal Policy Optimization framework, we enable reversible access to the Markov Decision Process dynamics and help the PPO to better estimate vehicle dynamics used for controller. A numerical simulation is conducted to demonstrate the trajectory following accuracy of the proposed methodology compared with popular industry control methods such as linear–quadratic regulator. Our results indicate that the proposed hybrid trajectory model based approach not only reduces the need for extensive data collection but also achieves similar control accuracy, suggesting a promising direction for future research in autonomous vehicle control, emphasizing the need for efficient, adaptable, and robust learning algorithms.

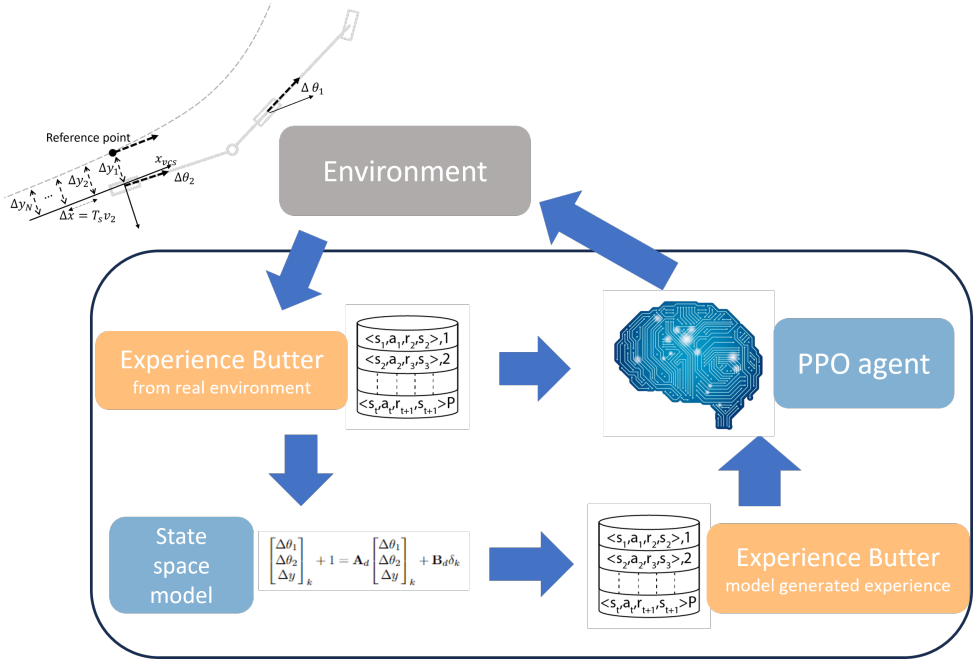


Figure 5.14: The MBRl structure with Dubins trajectory model generator

While model-free deep reinforcement learning has achieved remarkable successes across various domains, its application remains limited by substantial sample inefficiency and a lack of interpretability. Model-based reinforcement learning algorithms are generally regarded as being more efficient, the model-based policy search methods tackle the challenge of sample inefficiency by utilizing observed trajectories to construct a predictive model of the robot's dynamics and interactions with its surroundings [51]. However, previous works have relied on straightforward function approximators [52] or Bayesian models [53] to maintain sample efficiency and prevent overfitting. The application of these methods to diverse and intricate high-dimensional tasks is challenging. Efforts to address these issues have included the use of large-scale neural networks to capture the intricate dynamics typical of deep reinforcement learning benchmarks. However, these models often require extended training periods and struggle with transferability and interpretability [54], which confines the application scope of these models to tasks with relatively static scenarios and assignments.

By integrating the Dubin path planning with preview control, we build the system dynamics model to represent the environment interaction, where the preview distance control is also applied. In Figure 5.15, the first N preview reference points and the deviation value are plotted within the trailer VCS, and the reference error value for each preview reference sample is computed and denoted by $(\theta_{1_k}, \theta_{2_k}, y_k)$, Δx is the sampling interval and T_s is the simulation step time. Therefor, the N preview sample points can be computed by:

$$\begin{bmatrix} \Delta\theta_1 \\ \Delta\theta_2 \\ \Delta y \end{bmatrix}_k + 1 = \mathbf{A}_d \begin{bmatrix} \Delta\theta_1 \\ \Delta\theta_2 \\ \Delta y \end{bmatrix}_k + \mathbf{B}_d \delta_k \tag{5.30}$$

where

$$A_r = \begin{bmatrix} 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 1 \\ 0 & 0 & 0 & \cdots & 0 \end{bmatrix}_{(N \times N)}$$

and

$$B_r = \begin{bmatrix} 0 \\ 0 \\ 0 \\ \vdots \\ 1 \end{bmatrix}_{(N \times 1)}$$

We evaluate the effectiveness of the proposed learning strategy using the benchmark platform across various vehicle setups for trajectory tracking tasks. The initialization of the trajectory planning and motion prediction models were based on the kinematic state space model of the vehicles which requires a single calibration for each agent. Subsequently, these models networks can be transferred and re-trained during operation to manage distinct tasks for changing vehicle configurations. The adaptability of our method was evident as it successfully handled new challenges during testing, such as managing trailers with increased overhang length which introduced more intricate dynamics.

Figure 5.16 compares the mean episode lengths for PPO-trained agents with 2-meter and 4-meter trailers as well as an RPPO-trained agent trained with 2-meter trailer. The PPO agent trained with a shorter trailer shows better performance, achieving efficient parking after 250,000 training steps. In contrast, the longer trailer brings more complexity

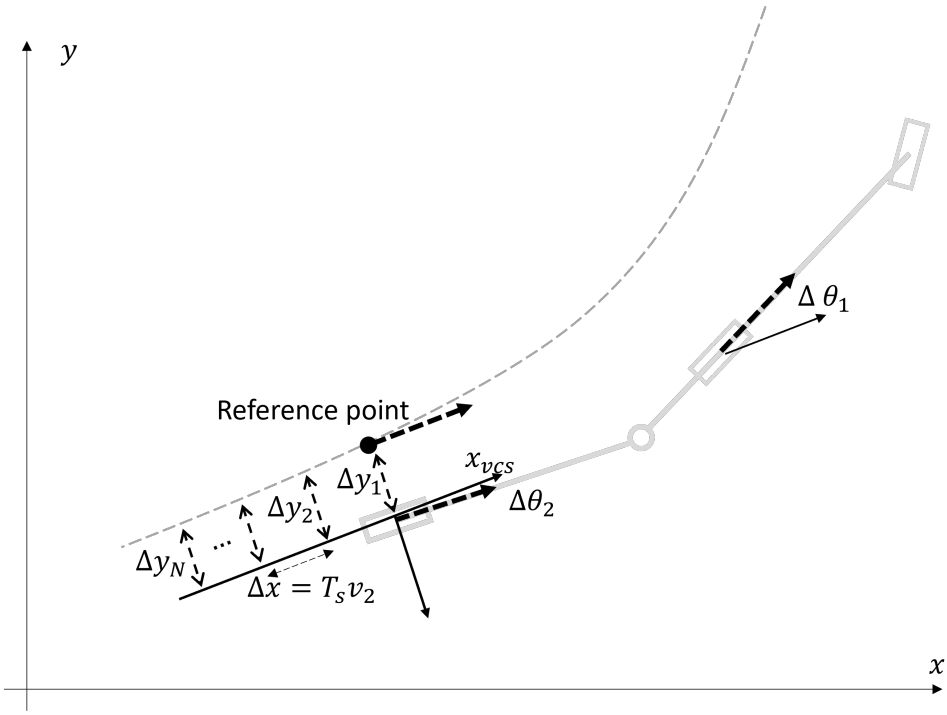


Figure 5.15: Vehicle kinematics model with reference

for reverse driving maneuvering, which leads to a slower training, although it achieves similar parking success rate and rewards in 5.17. However, the success rate of RPPO algorithm is under 50% and the training takes more steps, possibly due to vanishing gradients and the intricate nature of the task's delayed rewards.

Figure 5.17 shows that the PPO with a 2-meters trailer is able to achieving stable rewards by 250,000 steps. The training of 4m trailer's catches up by 300k steps, reflecting the added complexity in controlling longer trailers and the consequent need for extended training to refine the parking maneuvers.

Figure 5.18 compares the LQR controller with the proposed algorithm regarding truck orientation error, trailer orientation error, and lateral offset error. For a 2-meter trailer which is popular in North America market which handles most moving tasks, the performance of both systems is closely matched, with the lateral offset consistently maintained below 0.2 meters. The result shows the proposed algorithm's high tracking accuracy and suggests its potential effectiveness in practical scenarios.

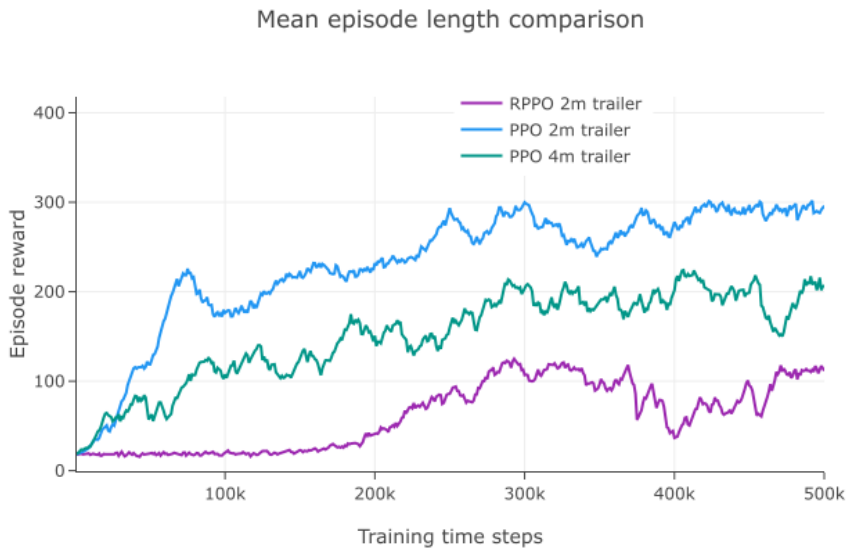


Figure 5.16: Mean episode length comparison between RPPO training on 2 meters trailer, PPO training on both 2 meters and 4 meters trailer

Figure 5.17: Mean episode training reward comparison between PPO training on both 2 meters and 4 meters trailer

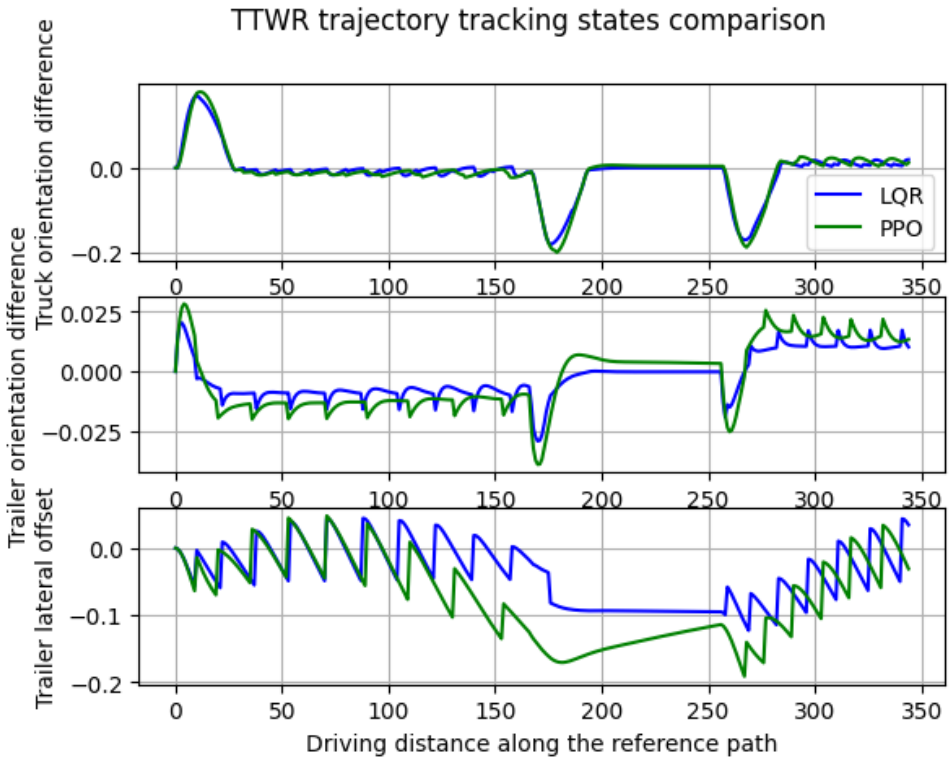


Figure 5.18: The comparison of LQR controller and proposed algorithm controlling normal 2m trailer v.s. reference trajectory orientation error, trailer orientation error, and trailer lateral offset error

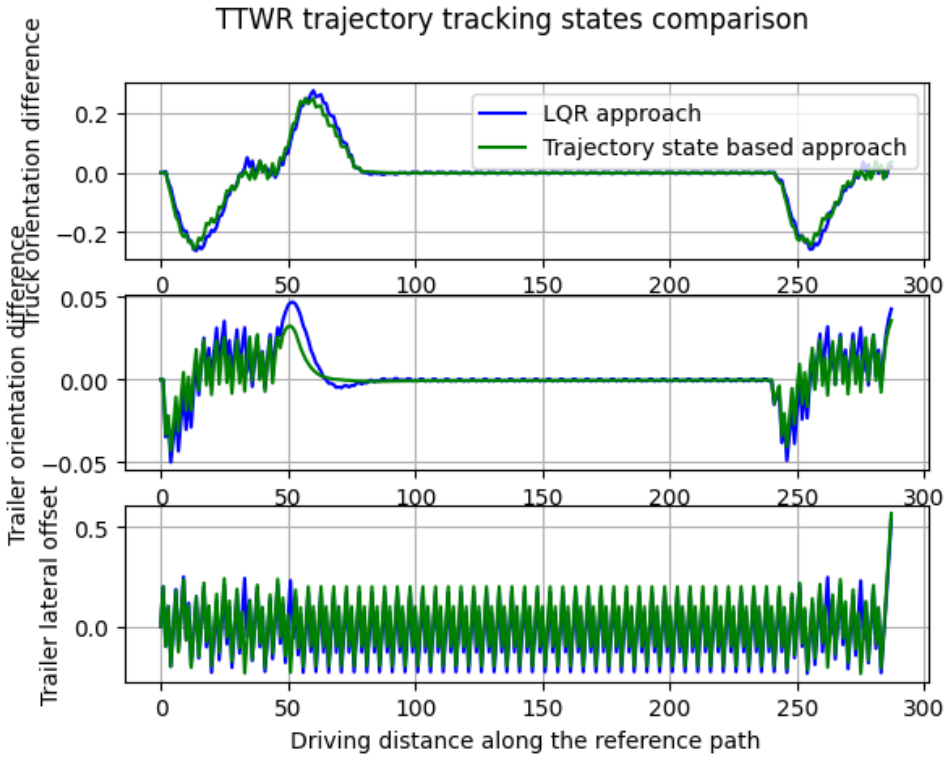


Figure 5.19: The comparison of LQR controller and proposed algorithm controlling extended trailer v.s. reference trajectory orientation error, trailer orientation error, and trailer lateral offset error

Figure 5.19 presents a comparative analysis of the LQR controller and the proposed algorithm, applied to a 5-meter extended trailer designed for heavy-duty tasks. Both controllers successfully follow the given trajectory; however, the proposed algorithm demonstrates slightly better performance in orientation tracking with smaller errors. This indicates enhanced adaptability of the proposed algorithm in contrast to the LQR controller, which relies on fixed feedback gains.

Chapter 6

Dissertation Work Plan

6.1 Dissertation Progresses Work

Phase 1: Preliminary Analysis and Model Development

1. **Literature Review**:
    - Study existing literature on truck-trailer dynamics, reverse parking algorithms, and model-based reinforcement learning applications in vehicle control.
    - Identify gaps in current methodologies and potential improvements.
2. **Data Collection**:
    - Gather data on truck-trailer dynamics during reverse parking maneuvers. This can include sensor data, camera feeds, and other relevant telemetry.
    - Use simulators if real-world data collection is challenging or risky.
3. **Model Development**:
    - Develop a dynamic model of the truck-trailer system. This model should capture the kinematics and dynamics of reverse parking.

- Validate the model using collected data or through simulation environments.

Phase 2: Reinforcement Learning Framework and Simulation

1. **RL Environment Setup**:
    - Define the state space (positions, angles, velocities), action space (steering angles, acceleration), and reward function (based on parking accuracy, safety, and maneuver time).
    - Set up terminal conditions, such as successful parking or collision scenarios.
2. **Algorithm Selection and Development**:
    - Choose an appropriate model-based RL algorithm. Consider algorithms that can efficiently utilize the truck-trailer model to simulate future trajectories.
    - Implement the algorithm and integrate it with the dynamic model.
3. **Simulation and Testing**:
    - Use a high-fidelity simulator to test the RL agent's performance. Ensure the simulator can accurately replicate real-world parking scenarios.
    - Iteratively train the agent, refining the model and RL algorithm based on simulation results.

Phase 3: Comparative Analysis with Other Control Algorithms

1. **Selection of Control Algorithms**:

- Identify other prominent control algorithms used for truck-trailer reverse parking, such as LQR, MPC, or other traditional methods.

2. **Implementation and Testing**:
   - Implement the selected control algorithms in the simulation environment.
   - Test each algorithm under similar conditions to ensure a fair comparison.

3. **Performance Evaluation**:
   - Evaluate the performance of each algorithm based on criteria like parking accuracy, safety, time taken, and control smoothness.
   - Compare the results of the model-based RL approach with the results of the other algorithms.

4. **Analysis and Reporting**:
   - Analyze the strengths and weaknesses of the model-based RL approach in comparison to the other algorithms.
   - Document findings, insights, and potential areas for improvement or further research.

6.2 Deliverables

This research aims to explore the potential of model-based reinforcement learning (MBRL) in the domain of truck-trailer reverse parking control. The objective is to develop an efficient and robust control strategy that can outperform traditional methods. The deliverables outlined below provide a comprehensive overview of the expected outcomes, tools, and documentation that will be produced during the research process.

1. **Dynamic Model of the Truck-Trailer System**:

- A mathematical representation capturing the kinematics and dynamics of the truck-trailer during reverse parking maneuvers.
- Validation results comparing the model's predictions with simulation or real-world data.

2. **Benchmark Control Algorithms based on Trarditional Algorithms**:
   - Dubins path planning for reverse parking path generation
   - A* path planning for spot search and trajectory generation
   - LQR controller
   - MPC controller

3. **Reinforcement Learning Framework**:
   - Detailed documentation of the chosen DQN RL algorithm, including its advantages and limitations.
   - Detailed documentation of the chosen PPO RL algorithm, including its advantages and limitations.
   - Detailed documentation of the chosen model-based RL algorithm, including its advantages and limitations.
   - Source code of the implemented RL algorithm, accompanied by comments and usage instructions.

4. **Simulation Results**:
   - A comprehensive report detailing the performance of the RL agent in various simulated scenarios.
   - Visual aids such as graphs, plots, and possibly videos showcasing the agent's performance.

5. **Comparative Analysis**:

    - A structured comparison between the MBRL approach and other control algorithms.
    - Performance metrics, challenges faced, and potential areas of improvement for each method.

6. **Software Tools and Utilities**:

    - Any custom software or utilities developed during the research, including simulation tools, data processing scripts, and visualization aids.
    - Documentation and user manuals for these tools, ensuring future researchers or collaborators can utilize them effectively.

7. **Final Thesis**:

    - A comprehensive document detailing the entire research process, methodologies, results, and conclusions.
    - Recommendations for future research or potential real-world applications of the developed MBRL approach.

6.3 Research Schedule

The following is the coursework layout and dissertation schedule separated by milestones and months:

1. **Model Development and Validation (September 2023 - December 2023)**

    - Develop and validate the dynamic model of the truck-trailer system.
    - Begin preliminary work on the benchmark control algorithms based on traditional methods.

2. **Reinforcement Learning Framework Development (November 2023 - March 2024)**

    - Implement and test the chosen DQN, TD3, and model-based RL algorithms.
    - Start simulations and gather initial results.

3. **Comparative Analysis and Refinement (April 2024 - May 2024)**

    - Conduct a structured comparison between the MBRL approach and other control algorithms.
    - Refine and optimize the RL algorithms based on the comparative analysis.

4. **Finalization and Thesis Writing (July 2024 - August 2024)**

    - Finalize all research components, including software tools and utilities.
    - Write, review, and finalize the PhD thesis.

References

[1] B. Koenig, *With Heavy Vehicles, Self-Driving Is Old Hat - The Robotics Institute Carnegie Mellon University*. [Online]. Available: `https://www.ri.cmu.edu/heavy-vehicles-self-driving-old-hat/` (visited on 08/06/2023).

[2] J. Chen, B. Yuan, and M. Tomizuka, "Model-free deep reinforcement learning for urban autonomous driving," in *2019 IEEE intelligent transportation systems conference (ITSC)*, IEEE, 2019, pp. 2765–2771.

[3] A. Shaout, A. Shaout, and S. A. Varagiri, "Adas technology classifications using a modified agile process," in *2022 International Arab Conference on Information Technology (ACIT)*, IEEE, 2022, pp. 1–7.

[4] Q. Liu, X. Li, Y. Zhu, and S. Li, "A review of tracking control method for tractor-trailer vehicles," in *Proceedings of the 2022 2nd International Conference on Control and Intelligent Robotics*, 2022, pp. 863–867.

[5] J. Jing, J. M. Maroli, Y. B. Salamah, M. Hejase, L. Fiorentini, and Ü. Özgüner, "Control method designs and comparisons for tractor-trailer vehicle backward path tracking," in *2019 American Control Conference (ACC)*, IEEE, 2019, pp. 5531–5537.

[6] H. Xu, Z. Yan, J. Xuan, G. Zhang, and J. Lu, "Improving proximal policy optimization with alpha divergence," *Neurocomputing*, vol. 534, pp. 94–105, 2023.

[7] M. Zanchetta, D. Tavernini, A. Sorniotti, *et al.*, "Trailer control through vehicle yaw moment control: Theoretical analysis and experimental assessment," *Mechatronics*, vol. 64, p. 102 282, 2019.

[8] F. Ye, X. Cheng, P. Wang, C.-Y. Chan, and J. Zhang, "Automated lane change strategy using proximal policy optimization-based deep reinforcement learning," in *2020 IEEE Intelligent Vehicles Symposium (IV)*, IEEE, 2020, pp. 1746–1752.

[9] S. L. Brunton and J. N. Kutz, *Data-driven science and engineering: Machine learning, dynamical systems, and control*. Cambridge University Press, 2022.

[10] C. Chen, Y. Liu, S. Kreiss, and A. Alahi, "Crowd-robot interaction: Crowd-aware robot navigation with attention-based deep reinforcement learning," in *2019 international conference on robotics and automation (ICRA)*, IEEE, 2019, pp. 6015–6022.

[11] Z. Du, Q. Miao, and C. Zong, "Trajectory planning for automated parking systems using deep reinforcement learning," *International Journal of Automotive Technology*, vol. 21, pp. 881–887, 2020.

[12] M. Jaritz, R. De Charette, M. Toromanoff, E. Perot, and F. Nashashibi, "End-to-end race driving with deep reinforcement learning," in *2018 IEEE international conference on robotics and automation (ICRA)*, IEEE, 2018, pp. 2070–2075.

[13] Q. Wang, J. Cheng, and H. Zhang, "Policy gradient reinforcement learning method for backward motion control of tractor-trailer mobile robot," in *Proceedings of the 11th International Conference on Computer Engineering and Networks*, Springer, 2021, pp. 303–311.

[14] E. Bejar and A. Moran, "A preview neuro-fuzzy controller based on deep reinforcement learning for backing up a truck-trailer vehicle," in *2019 IEEE Canadian Conference of Electrical and Computer Engineering (CCECE)*, IEEE, 2019, pp. 1–4.

[15] M. A. Sprayberry, "A data-driven approach to process parameter optimization in additive manufacturing via electron beam melting," Ph.D. dissertation, Northcentral University, 2023.

[16] M. Buehler, K. Iagnemma, and S. Singh, *The 2005 DARPA grand challenge: the great robot race*. springer, 2007, vol. 36.

[17] M. Buehler, K. Iagnemma, and S. Singh, *The DARPA urban challenge: autonomous vehicles in city traffic*. springer, 2009, vol. 56.

[18] J. Reeds and L. Shepp, "Optimal paths for a car that goes both forwards and backwards," *Pacific journal of mathematics*, vol. 145, no. 2, pp. 367–393, 1990.

[19] E. W. Dijkstra, "A note on two problems in connexion with graphs," in *Edsger Wybe Dijkstra: His Life, Work, and Legacy*, 2022, pp. 287–290.

[20] P. E. Hart, N. J. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *IEEE transactions on Systems Science and Cybernetics*, vol. 4, no. 2, pp. 100–107, 1968.

[21] T. Dudi, R. Singhal, and R. Kumar, "Shortest path evaluation with enhanced linear graph and dijkstra algorithm," in *2020 59th Annual Conference of the Society of Instrument and Control Engineers of Japan (SICE)*, IEEE, 2020, pp. 451–456.

[22] A. Stentz, "Optimal and efficient path planning for partially-known environments," in *Proceedings of the 1994 IEEE international conference on robotics and automation*, IEEE, 1994, pp. 3310–3317.

[23] S. LaValle, "Rapidly-exploring random trees: A new tool for path planning," *Research Report 9811*, 1998.

[24] A. Ammar, H. Bennaceur, I. Châari, A. Koubâa, and M. Alajlan, "Relaxed dijkstra and a* with linear complexity for robot path planning problems in large-scale grid environments," *Soft Computing*, vol. 20, pp. 4149–4171, 2016.

[25] T. McMahon, A. Sivaramakrishnan, E. Granados, K. E. Bekris, *et al.*, "A survey on the integration of machine learning with sampling-based motion planning," *Foundations and Trends® in Robotics*, vol. 9, no. 4, pp. 266–327, 2022.

[26] [Online]. Available: `https://engineeringmedia.com/map-of-control`.

[27] S. Bennett, "A brief history of automatic control," *IEEE Control Systems Magazine*, vol. 16, no. 3, pp. 17–25, 1996.

[28] P. Saraf, M. Gupta, and A. M. Parimi, "A comparative study between a classical and optimal controller for a quadrotor," in *2020 IEEE 17th India Council International Conference (INDICON)*, IEEE, 2020, pp. 1–6.

[29] C. Urmson, J. Anhalt, D. Bagnell, *et al.*, "Autonomous driving in urban environments: Boss and the urban challenge," *Journal of field Robotics*, vol. 25, no. 8, pp. 425–466, 2008.

[30] M. Buehler, K. Iagnemma, and S. Singh, "Junior: The stanford entry in the urban challenge," *The DARPA urban challenge: autonomous vehicles in city traffic*, vol. 56, pp. 91–123, 2009.

[31] M. Bojarski, D. Del Testa, D. Dworakowski, *et al.*, "End to end learning for self-driving cars," *arXiv preprint arXiv:1604.07316*, 2016.

[32] A. Kendall, J. Hawke, D. Janz, *et al.*, "Learning to drive in a day," in *2019 International Conference on Robotics and Automation (ICRA)*, IEEE, 2019, pp. 8248–8254.

[33] B. Peng, Q. Sun, S. E. Li, *et al.*, "End-to-end autonomous driving through dueling double deep q-network," *Automotive Innovation*, vol. 4, pp. 328–337, 2021.

[34] W. B. Knox, A. Allievi, H. Banzhaf, F. Schmitt, and P. Stone, "Reward (mis) design for autonomous driving," *Artificial Intelligence*, vol. 316, p. 103 829, 2023.

[35] L. Chen, P. Wu, K. Chitta, B. Jaeger, A. Geiger, and H. Li, "End-to-end autonomous driving: Challenges and frontiers," *arXiv preprint arXiv:2306.16927*, 2023.

[36] Z. Huang, L. Liang, Z. Ling, X. Li, C. Gan, and H. Su, "Reparameterized policy learning for multimodal trajectory optimization," 2023.

[37] A. Hu, G. Corrado, N. Griffiths, *et al.*, "Model-based imitation learning for urban driving," *Advances in Neural Information Processing Systems*, vol. 35, pp. 20 703–20 716, 2022.

[38] F. Sorge, "Motion and force analysis of slow-speed multi-trailer systems," *Proceedings of the Institution of Mechanical Engineers, Part C: Journal of Mechanical Engineering Science*, vol. 233, no. 8, pp. 2615–2635, 2019.

[39] X. Liu, A. K. Madhusudhanan, and D. Cebon, "Minimum swept-path control for autonomous reversing of a tractor semi-trailer," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 5, pp. 4367–4376, 2019.

[40] G. Lei and Y. Zheng, "Research on cooperative trajectory planning algorithm based on tractor-trailer wheeled robot," *IEEE Access*, vol. 10, pp. 64 209–64 221, 2021.

[41] P. Polack, F. Altché, B. d'Andréa Novel, and A. de La Fortelle, "The kinematic bicycle model: A consistent model for planning feasible trajectories for autonomous vehicles?" In *2017 IEEE intelligent vehicles symposium (IV)*, IEEE, 2017, pp. 812–818.

[42] S. Abraham and J. Nilsson, "Trailer parking assist," 2013.

[43] R. W. Brockett, *Finite dimensional linear systems*. SIAM, 2015.

[44] C. Altafini, A. Speranzon, and B. Wahlberg, "A feedback control scheme for reversing a truck and trailer vehicle," *IEEE Transactions on robotics and automation*, vol. 17, no. 6, pp. 915–922, 2001.

[45] C. Altafini, "Controllability and singularities in the n-trailer system with kingpin hitching," *IFAC Proceedings Volumes*, vol. 32, no. 2, pp. 2169–2174, 1999.

[46] L. E. Dubins, "On curves of minimal length with a constraint on average curvature, and with prescribed initial and terminal positions and tangents," *American Journal of mathematics*, vol. 79, no. 3, pp. 497–516, 1957.

[47] N. Lehtomaki, N. Sandell, and M. Athans, "Robustness results in linear-quadratic gaussian based multivariable control designs," *IEEE Transactions on Automatic Control*, vol. 26, no. 1, pp. 75–93, 1981.

[48] M. Schwenzer, M. Ay, T. Bergs, and D. Abel, "Review on model predictive control: An engineering perspective," *The International Journal of Advanced Manufacturing Technology*, vol. 117, no. 5-6, pp. 1327–1349, 2021.

[49] H. Yan, M. A. Zohdy, A. Shaout, and A. Mahmoud, "Recurrent proximal policy optimization based tractor-trailer wheeled robot automatic parking algorithm," *Transactions on Engineering and Computing Sciences*, vol. 11, no. 4, 124–142, 2023. DOI: 10.14738/tecs.114.15355. [Online]. Available: https://journals.scholarpublishing.org/index.php/TMLAI/article/view/15355.

[50] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.

[51] J. Schulman, S. Levine, P. Abbeel, M. Jordan, and P. Moritz, "Trust region policy optimization," in *International conference on machine learning*, PMLR, 2015, pp. 1889–1897.

[52] R. Lioutikov, A. Paraschos, J. Peters, and G. Neumann, "Sample-based informationl-theoretic stochastic optimal control," in *2014 IEEE international conference on robotics and automation (ICRA)*, IEEE, 2014, pp. 3896–3902.

[53] M. Deisenroth and C. E. Rasmussen, "Pilco: A model-based and data-efficient approach to policy search," in *Proceedings of the 28th International Conference on machine learning (ICML-11)*, 2011, pp. 465–472.

[54] A. Nagabandi, G. Kahn, R. S. Fearing, and S. Levine, "Neural network dynamics for model-based deep reinforcement learning with model-free fine-tuning," in *2018 IEEE international conference on robotics and automation (ICRA)*, IEEE, 2018, pp. 7559–7566.