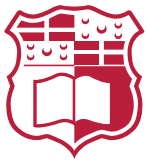
Reinforcement Learning for Autonomous Navigation of Articulated Vehicles

Daniel Attard

Supervisor: Dr. Josef Bajada

November 2023

Submitted in partial fulfilment of the requirements
for the degree of Master of Science in Artificial Intelligence.



L-Università ta' Malta
Faculty of Information &
Communication Technology

Abstract

This research delves into the development of a reinforcement learning based model for the lateral and longitudinal control of tractor-trailer vehicles through a roundabout intersection. With such vehicles being a crucial part of the logistics sector, which provides basic needs to our society, the development of autonomous driving models for such vehicles has the potential to reduce the costs of the logistics sector through greater fuel and time efficiency and reduce accidents caused by human fatigue.

Despite this, so far, limited research that covers the motion control of articulated heavy goods vehicles has been carried out. Due to their physical properties, tractor-trailer vehicles require specially crafted models which obtain a suitable representation of their movement. Such movement differs from light passenger vehicles due to the articulated joint, length, width, height, and weight of tractor-trailer vehicles. Previous researchers developing rule-based approaches note that the mathematical models used to define the dynamics models of the tractor-trailer vehicle, which are the basis of their work, are simplified for the benefit of computational complexity. This limits the applicability of rule-based models in different environments without specific fine-tuning.

Reinforcement learning allows the development of models which learn through trial and error, enabling them to obtain superior performance to rule-based models in unknown environments. Firstly, due to the lack of publicly available models, this work proposes a model of a tractor-trailer vehicle in the high-fidelity CAR Learning to Act (CARLA) simulator. Five different roundabout scenarios were also developed to perform the driving trials. Such intersections have varying physical properties, allowing for the discovery of each model's advantages and pitfalls. Being the first of its kind, this work proposes a reinforcement learning environment which can be utilised for such vehicles. The 69 continuous observations provide information about the vehicle and the route.

We compare the PPO, dueling double DQN, and SAC algorithms, with the on-policy, policy optimisation algorithm, PPO, obtaining superior results with a 0.77 success rate on testing scenarios while also maintaining the least distance to the centre of the lane, mimicking human-like behaviour. Aiming to discover the full potential of this framework, the proportional-integral controller maintaining a constant velocity was eliminated with the reinforcement learning algorithm being able to accurately control the tractor-trailer vehicle's longitudinal and lateral movements, obtaining a 0.68 testing success rate using the PPO algorithm.

Acknowledgements

I would like to express my deepest and sincere gratitude to my supervisor, Dr Josef Bajada for his patience, support, understanding, and guidance throughout this research project. His encouragement, positive outlook, and invaluable advice made this dissertation possible.

I am also extremely grateful to my family and friends, for their encouragement and support throughout my studies.

I would also like to thank the Malta Digital Innovation Authority (MDIA) for believing in the importance of research related to artificial intelligence and providing me with the AI Scholarship to pursue my postgraduate studies.

Publications

Part of this work has been presented at the 'Trustworthy AI for safe & secure traffic control in connected & autonomous vehicles' (TACTFUL) workshop at the European Conference for Artificial Intelligence (ECAI) 2023 in Krakow, Poland:

D. Attard and J. Bajada, "Autonomous Navigation of Tractor-Trailer Vehicles through Roundabout Intersections," presented at the Trustworthy AI for safe & secure traffic control in connected & autonomous vehicles (TACTFUL) workshop at the European Conference for Artificial Intelligence (ECAI), Poland, October 2023.

Contents

List of Figures

List of Tables

List of Abbreviations

A3C Asynchronous Advantage Actor Critic.

API Application Programming Interface.

CARLA CAR Learning to Act.

CNN Convolutional Neural Network.

DDDQN Dueling Double Deep Q-Network.

DDPG Deep Deterministic Policy Gradient.

DDQN Double Deep Q-Network.

DQN Deep Q-Network.

DRL Deep Reinforcement Learning.

GRU Gated Recurrent Unit.

IDM Intelligent Driver Model.

IL Imitation Learning.

IMU Inertial Measurement Unit.

LGSVL LG Silicon Valley Lab.

LIDAR Light Detection and Ranging.

MBVE Model-Based Value Expansion.

MDP Markov Decision Process.

MOBIL Minimizing Overall Braking Induced by Lane Changes.

PI Proportional–Integral.

PID Proportional–Integral–Derivative.

PPO Proximal Policy Optimization.

RL Reinforcement Learning.

RNN Recurrent Neural Network.

ROS Robot Operating System.

SAC Soft Actor-Critic.

SUMO Simulation of Urban MObility.

TD3 Twin Delayed DDPG.

TRPO Trust Region Policy Optimization.

TTC Time-To-Collision.

1 Introduction

1.1 Motivation

Road traffic accidents lead to the death of 1.3 million people every year while an additional 20 to 50 million suffer from non-fatal injuries [1]. In the United States of America, 96% of traffic accidents are caused by human error [2]. Taking the human driver out of the equation can therefore significantly decrease such fatalities. The introduction of driver assistance technologies such as adaptive cruise control, traffic jam assistant, and collision avoidance systems, has already proved the potential of autonomous vehicles, reducing road fatalities in Europe by 48% in a span of 14 years [3]. Furthermore, autonomous vehicles help reduce congestion and increase mobility.

This work will delve into the autonomous navigation of heavy goods vehicles, specifically tractor-trailer vehicles, which are a crucial part of the logistics sector and make up 9.2% is the total distance driven [4]. Apart from the above-mentioned advantages, autonomous heavy goods vehicles can benefit from higher fuel efficiencies. Autonomous vehicles are able to control the vehicle more accurately, improving fuel efficiency and reducing carbon emissions, while also increasing the adoption of truck platooning [5]. Due to reduced air resistance, truck platooning, a number of tractor-trailer vehicles closely following one another, can reduce fuel consumption by 15% [6].

Being a vital component of how society is able to obtain its products and services, the shortage of tractor-trailer vehicle drivers is of major concern [7]. Further to this, such human drivers are bound by sleeping schedules which reduce the efficiency of the logistics sector. Autonomous heavy goods vehicles can alleviate these issues while also reducing costs and monotonous jobs. Despite such advantages, research related to the autonomous navigation of heavy goods vehicles is relatively sparse and due to their unique physical characteristics and dynamic systems, models developed for light passenger vehicles may generate undesirable behaviour if applied to such vehicles.

1.2 Problem Definition

A tractor-trailer vehicle is defined as two rigid units, the tractor unit and trailer unit, connected together using an articulated pivot. Such vehicles are significantly heavier, higher, and longer than normal passenger vehicles. The combination of the articulated pivot and these physical properties introduces some undesirable behaviour. Firstly, the higher centre of gravity increases the chances of such vehicles rolling over at higher speeds and steering angles. Secondly, when turning at lower speeds, the rear trailer wheels follow a tighter radius path than the front wheels of the truck [8]. The causes of

such undesirable properties are prevalent in roundabout intersections where tractor-trailer vehicles may need to navigate through small-diameter roundabouts at slower speeds and large-diameter roundabouts at higher speeds. Therefore, in order to avoid any collisions, the system must deviate the tractor unit away from the centre of the lane on smaller diameter roundabouts and limit the maximum velocity on larger diameter roundabouts.

This research will focus on the autonomous navigation of tractor-trailer vehicles through roundabout intersections. Roundabout intersections are abundant in current road networks and present specific challenges due to their varying physical properties and high interaction between road users.

1.3 Overview of Current Techniques

Autonomous vehicle navigation is composed of four basic functions: localisation, perception, planning, and control [9]. While some research focuses on end-to-end systems [10, 11], resolving all four functions, other research focuses on solving one specific function [12, 13]. Furthermore, autonomous driving techniques can be categorised into rule-based and learning-based methods.

Rule-based methods benefit from consistent and explainable outputs. Model predictive control and sequential quadratic programming have been utilised for the development of a trajectory planner for tractor-trailer vehicles [14, 15]. At a lower level, Proportional–Integral–Derivative (PID) controllers and pure-pursuit algorithms have been used to control the longitudinal, acceleration, and lateral, steering, controls of the vehicle [16, 17]. Despite their reliability in simple environments, rule-based methods lack accuracy in more complex environments, similar to real-world scenarios [18]. Furthermore, the mathematical models used to represent vehicles may be highly simplified in favour of computational complexity, leading to inaccurate results [14]. This is especially true for the more complex physical system of tractor-trailer vehicles.

Learning-based approaches have seen significant improvements in recent years due to recent developments in Imitation Learning (IL) and Deep Reinforcement Learning (DRL). Such approaches require large amounts of data to train on. Despite this, IL, which imitates the behaviour of an expert, has achieved high success rates in the field of autonomous navigation [19].

RL approaches learn by trial and error through interaction with an environment. Such approaches are able to perform suitably in unknown and challenging environments and can surpass human-level performance [20]. Several previous research has focused on using DRL for autonomous navigation. Roundabout insertion focuses on choosing the right moment to insert a vehicle into a roundabout with circulating traffic, increasing

the traffic flow [12, 21]. This can also be achieved by modelling driver behaviour [22], including that of manually driven vehicles. In a multi-lane roundabout, lane changing with traffic can produce difficult situations where multiple vehicles and their behaviour have to be analysed [23]. End-to-end systems with throttle control [19] and steering control [24] have also been successfully developed. Path-tracking algorithms [16, 17] have also been successfully implemented. Inputs to such algorithms also vary from simple vehicle properties, such as velocity and position [25], to Time-To-Collision (TTC), radius of roundabout [13], and sensor-based inputs such as Light Detection and Ranging (LIDAR) and vision cameras [24]. Furthermore, most types of RL algorithms have been tested including q-learning [25, 26], policy optimization [12, 16] and actor-critic [13, 27] algorithms.

Systems that perform lane changing [28], reversing [29], tractor headland turning [30], and single-lane roundabout navigation [25], were also developed for tractor-trailer vehicles.

1.4 Aim and Objectives

The aim of the research is to develop a system to navigate a tractor-trailer vehicle through a multi-lane roundabout intersection. To reach this aim, the following objectives have been set:

1. Produce a tractor-trailer vehicle model, and roundabout scenarios, compatible with a high-fidelity autonomous driving simulator.
2. Develop a reinforcement learning environment to navigate a tractor-trailer vehicle through a multi-lane roundabout intersection, using vehicle and route information, distance sensors, and steering actions.
3. Compare different state-of-the-art RL algorithms such as PPO, DDDQN, and SAC.
4. Expand the policy optimisation algorithm to incorporate acceleration actions.

1.5 Proposed Solution

The lack of work focusing on RL approaches for the autonomous navigation of tractor-trailer vehicles provides several research gaps that need to be addressed for further development of such vehicles. The purpose of this work is to develop an end-to-end longitudinal and lateral control approach to navigate a tractor-trailer vehicle through a roundabout intersection using an RL algorithm, where such research is the first of its kind. Due to the lack of available models, this work starts off with developing the tractor

and trailer units in the high-fidelity simulator CAR Learning to Act (CARLA). Secondly, a RL framework will be developed on which different variants and RL algorithms will be trained and tested. A quantitative evaluation illustrates that when using a PPO algorithm, a 0.77 success rate is obtained on testing roundabout scenarios while a qualitative analysis visually illustrates the human-like behaviour of the developed agent. Compared to DDDQN algorithm and a SAC algorithm, the PPO exhibits superior performance obtaining high success rates along with low deviations from the centre of the lane. A variable velocity based action space also showcases the potential of the developed PPO model to accurately control both the lateral and longitudinal movements of a tractor-trailer vehicle. Furthermore, when comparing to current literature, our models obtain higher success rates while the tractor and trailer units experience less deviation from the centre of the lane.

1.6 Document Structure

The rest of this document is structured as follows:

Chapter 2 Background discusses the basic principles of different components used throughout this work including articulated vehicle and reinforcement learning algorithms.

Chapter 3 Literature Review focuses on reviewing previous literature related to both light and heavy goods vehicles while also including rule-based and reinforcement learning-based approaches.

Chapter 4 Methodology describes the different components of the methodology used where justification is made for any decision taken.

Chapter 5 Evaluation qualitatively and quantitatively evaluates the results obtained and compares the developed models.

Chapter 6 Conclusion concludes this work, highlighting the key contributions of this work.

2 Background

This chapter introduces the main concepts that will be used throughout this work. Firstly, articulated vehicles and their specific physical properties will be discussed, followed by, the fundamental concepts of RL such as the Markov Decision Process. After analysing the advantages and disadvantages of different RL algorithms, the mechanics of the RL algorithms used in this work are described in detail.

2.1 Articulated Heavy Vehicles

Articulated heavy vehicles are composed of a number of rigid units connected together through pivots. A truck with one trailer attached is the most common combination, known as a tractor-trailer or an articulated lorry. A truck with multiple trailers attached is also used, but these are out of the scope of this study.

The pivot allows manoeuvring of tighter turns when compared to a single rigid vehicle of the same length yet it introduces some undesirable behaviour. A set of criteria were defined by the Australian National Transport Commission [31] to determine the safety of tractor-trailer vehicles. Among many, low-speed swept path, and steady-state rollover threshold are the criteria relevant to the manoeuvres performed in this research. The low-speed swept area criteria, highlights the fact that when turning at low speeds, the rear trailer wheels follow a tighter radius path than the front wheels of the tractor unit [8]. This is illustrated in Figure 2.1. To compensate for this, a larger radius turn has to be navigated by the tractor unit.

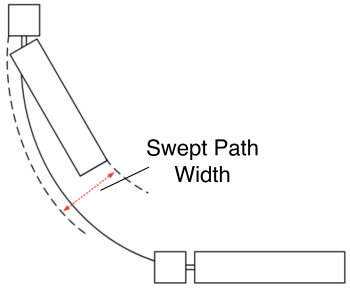


Figure 2.1 Low-speed swept area criteria [8]

The steady-state rollover threshold [8] is the maximum lateral acceleration a vehicle can withstand before rolling over in a turn. Due to their high centre of gravity, tractor-trailer vehicles are more prone to rollover while turning when compared to smaller passenger vehicles. Higher speeds and tighter turns decrease the lateral acceleration required for a vehicle to roll over.

2.2 Markov Decision Processes

A Markov Decision Process (MDP) is a mathematical framework used to specify a state-transition system, where the state-transitions are probabilistic and also carry rewards [32]. As illustrated in Equation 2.1, an MDP describes the probability of moving to state s' and awarded reward r, given that in the current state s, action a is taken.

$$p(s', r|s, a) \tag{2.1}$$

where $s', s \in S, r \in \mathbb{R}, a \in A$.

An MDP can be expressed as a 5-tuple $\langle S, A, R, f, \gamma \rangle$. The state space, S, represents information obtained from the environment, for example, the velocity of the vehicle. Applying throttle, longitudinal control, and changing the steering angle, lateral control, are possible actions in the action space, A. Applying the selected action to the current state of the environment results in a numerical reward, R, which can be composed of different components. The state-transition model, f, defines the effects of actions on state changes. The discount factor, γ, is the present value of future reward [32], where $0 \leq \gamma \leq 1$. When $\gamma = 0$, the agent is only concerned with maximising the current reward. This may restrict its access to future rewards, leading to a reduced overall return. When $\gamma = 1$, future rewards are given equal importance [32].

The system may also possess the Markov property. This is where the next state is only dependent on the current state and action, as shown by Equation 2.2 [33]. Any historical information that may affect future decisions is stored in the current state [32].

$$f(s_{t+1}|s_t, a_t, s_{t-1}, a_{t-1}, ..., s_0, a_0) = f(s_{t+1}|s_t, a_t) \tag{2.2}$$

2.3 Reinforcement Learning

RL is a subset of machine learning where an agent interacts with an environment, and an optimal policy that maximises the reward is found through trial and error [20]. Such algorithms have been successfully applied to various fields, such as energy [34], education [35], healthcare [36], and security [37]. RL algorithms are different from supervised and unsupervised learning, where the former maps inputs to labels while the latter tries to find hidden structure in the data [32].

RL can be used to solve an MDP, by finding an optimal policy. As shown in Figure 2.2, at each discrete time step t, the agent obtains the environment's state, s_t, from the set of all possible states, S, and chooses action, a_t, from the set of all possible actions, A, based on the current policy, $\pi(a_t|s_t)$. A reward, r_t, is given to the agent based on the reward function, $R(s_t, a_t, s_{t+1})$, and moves to the next state, s_{t+1}, based on state-

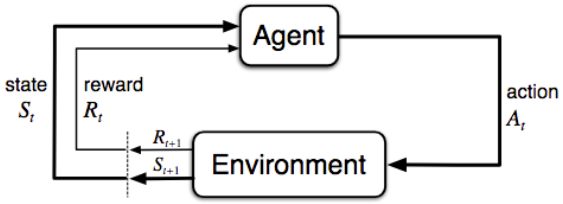


Figure 2.2 The Reinforcement Learning Interaction Loop [32]

transition model, $f(s_{t+1}|s_t, a_t)$. A discounted, accumulated reward is returned for each episode, shown in Equation 2.3 [20].

$$R_t = \sum_{k=0}^{\infty} \gamma^k r_{t+k} \tag{2.3}$$

A value function is used to predict the expected reward. Two value functions are used. The state-value function defined in Equation 2.4, is the expected return when following policy, π, from state, s [20].

$$v_\pi(s) = E[R_t|s_t = s] \tag{2.4}$$

This expands into the Bellman equation [20],

$$v_\pi(s) = \sum_a \pi(a|s) \sum_{s',r} p(s', r|s, a)[r + \gamma v_\pi(s')] \tag{2.5}$$

The action-value function defined in Equation 2.6, is the expected return when choosing an action, a, while in state, s, and then follow policyπ [20].

$$q_\pi(s, a) = E[R_t|s_t = s, a_t = a] \tag{2.6}$$

This expands into the Bellman equation [20],

$$q_\pi(s, a) = \sum_{s',r} p(s', r|s, a)[r + \gamma \sum_{a'} \pi(a'|s')q_\pi(s', a')] \tag{2.7}$$

The policy which obtains the highest return is the optimal policy, π^*. Such policies have optimal value functions, $v_*(s)$ and $q_*(s, a)$, which have the highest expected returns for all $s \in S$ and $a \in A$, as defined in Equations 2.8 and 2.9 [25].

$$v_*(s) = \max_\pi v_\pi(s), \forall s \in S \tag{2.8}$$

$$q_*(s, a) = \max_\pi q_\pi(s, a), \forall s \in S, \forall a \in A \tag{2.9}$$

2.3.1 On-policy vs Off-policy

In an attempt to handle a dilemma all learning methods experience, two types of RL methods are utilised [32]. A policy needs to follow non-optimal behaviour to explore and find the optimal actions while also learning action values based on the optimal behaviour. The policy aims to find a balance between exploring the environment and exploiting the current knowledge. On-policy methods balance this dilemma by following a near-optimal policy that still explores [32]. Off-policy methods utilise two policies, one of which learns the optimal policy while the other explores the environment, possibly discovering unknown optimal actions [32].

On-policy algorithms such as, Asynchronous Advantage Actor Critic (A3C) [38], Trust Region Policy Optimization (TRPO) [39], and PPO [40], are simpler to implement but are sample inefficient since they only utilise the collected data once. This increases the convergence time frame as the complexity of the task increases. Despite having slower convergence and increased variance, off-policy algorithms, including Deep Q-Network (DQN) [41] and SAC [42], are sample efficient and are able to learn certain, more complex, tasks [32]. Off-policy algorithms can also behave as on-policy algorithms by having both policies refer to the same policy [20].

2.3.2 Model-based vs Model-free

Model-based methods have knowledge of their environment and therefore can plan to obtain a value function or a policy. On the other hand, model-free approaches learn through trial and error in unknown environments. Model-based approaches such as, Model-Based Value Expansion (MBVE) [43], are data-efficient but inaccuracies in the utilised model may lead to sub-optimal results when applied in real-world scenarios. Model-free approaches, such as A3C [38] and SAC [42], permit the learning of complex problems whose environment cannot be easily represented [32].

2.3.3 Exploration vs Exploitation

A challenge in RL is balancing exploration and exploitation. While the agent can exploit its current knowledge to obtain the highest known reward, exploring different actions may result in discovering a better policy. Different exploration strategies are used to vary the balance between exploration and exploitation when training RL models. One strategy is using a stochastic sampling approach, which samples from a distribution. Another popular strategy is the epsilon greedy approach which chooses a random action with probability ϵ and the best action so far with probability $1 - \epsilon$ [20].

2.4 Deep Learning

Deep learning methods are representation learning methods which produce different levels of representation of the same data. This is achieved by performing a number of simple but non-linear transformations. This allows models to learn more complex functions without the need for domain expertise and careful engineering, which are required to obtain a suitable representation when using conventional machine-learning methods [44].

For example, when processing images, representations at initial layers include the presence or absence of edges. The following layers represent an increasingly larger group of edges and features extracted through kernel functions which, at the final layers, are able to detect objects [44].

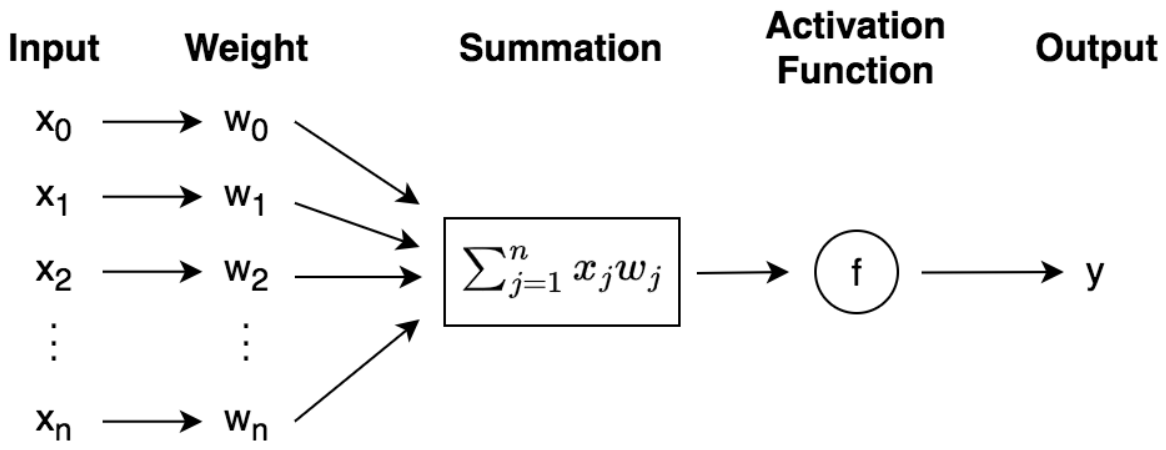


Figure 2.3 Artificial Neuron

In deep learning, one or more hidden layers are used between the input and output layers. Such layers are composed of a number of different artificial neurons. As depicted in Figure 2.3, the activation function computes the output value of the neuron based on the weighted sum of its inputs and a bias.

Figure 2.4 illustrates a neural network with 2 hidden layers. Error derivatives are computed and the weights and biases of each neuron are updated to minimise a loss function during a process called backpropagation, going from the output to the input layer. This determines the correct weights which map the input to the output values.

Deep learning methods have allowed Convolutional Neural Network (CNN) and Recurrent Neural Network (RNN) to achieve significant practical success in different fields. CNNs allow for traffic sign recognition [44], segmentation of biological images [44], and detection of faces [44], while RNNs are of benefit to the language [44] and time-series processing [45] fields.

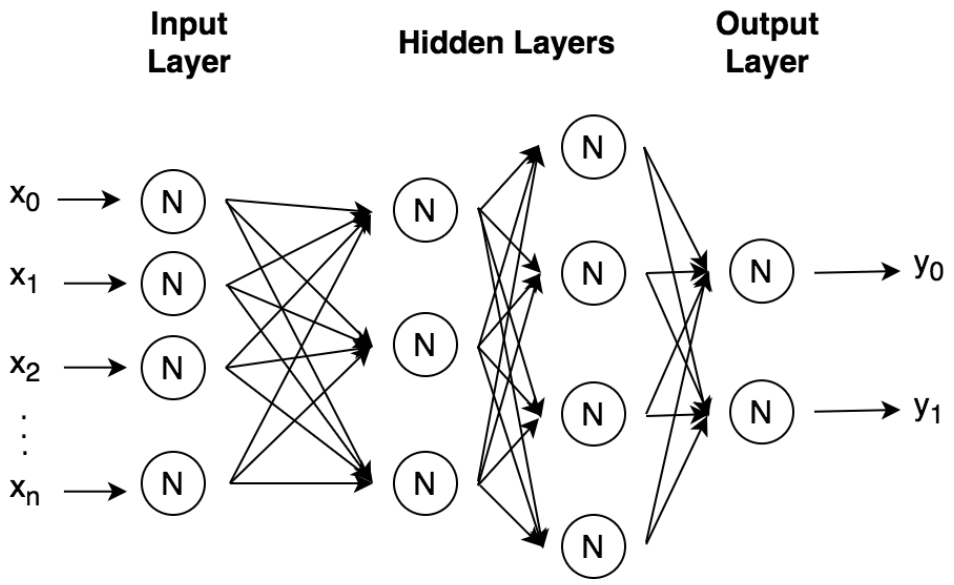


Figure 2.4 Neural Network

2.5 Deep Reinforcement Learning

DRL is the use of deep learning techniques to approximate one or more components of a reinforcement learning algorithm. These components include the value function, state-value, $\hat{v}(s;\theta)$, or action-value, $\hat{q}(s,a;\theta)$, the policy $\pi(a|s;\theta)$, and the state.

While RL is limited to low-dimensional problems and lacks scalability, DRL has been successfully applied to a multitude of different problems in various fields. DRL was used to play the game of Go, achieving and surpassing human-level performance [46]. More practical uses include the navigation of indoor spaces [47] and handling of power management and performance optimisation for computer systems [48].

2.5.1 Q-Learning

Deep Q-Network

Before DQN was proposed, RL was known to be unstable when a nonlinear function, like neural networks, was used to estimate the action value function [20]. DQN [41] made use of a number of different techniques to improve training stability, which led to achieving superior results compared to the current state-of-the-art algorithms on many Atari games [20].

Real-world problems include a large number of states and actions. This implies that learning each state-action value, as defined in Equation 2.6, is infeasible. To alleviate this problem, a DQN utilises a neural network to represent state-action values using a parameterised value function, $Q(s,a;\theta_t)$. The multi-layered neural network is a function of $\mathbb{R}^n \mapsto \mathbb{R}^m$, where n and m, are the state and action space dimensions.

The parameterised value function is updated using Equation 2.10.

$$\theta_{t+1} = \theta_t + \alpha(Y_t^Q - Q(S_t, A_t; \theta_t))\nabla_{\theta_t} Q(S_t, A_t; \theta_t) \tag{2.10}$$

where α is a scalar step size and Y_t^Q, the target value, is defined as,

$$Y_t^Q \equiv R_{t+1} + \gamma \max_a Q(S_{t+1}, a; \theta_t) \tag{2.11}$$

The current value function $Q(S_t, A_t; \theta_t))$ is updated towards Y_t^Q using stochastic gradient descent with learning rate α [49].

This target value function can be re-written as,

$$Y_t^Q = R_{t+1} + \gamma Q(S_{t+1}, \arg\max_a Q(S_{t+1}, a; \theta_t); \theta_t) \tag{2.12}$$

To increase training stability, DQN makes use of deep neural networks to approximate the action value function [20]. In order to smooth out data distribution changes and remove correlations in the data, a replay buffer is used. This allows the reuse of the same data sample over multiple epochs. This was further improved by the development of prioritised experience replay, where priority is given to each observation using a stochastic sampling method, and the observation is sampled according to its priority [20]. To reduce correlations between action values Q and the target $r + \gamma max_{a'} Q(s^{'}, a^{'})$, a separate network, called the target network, stores the network parameters which are only updated periodically. Furthermore, in order to improve stability, DQN reduces the input dimensionality and performs error clipping [20].

Double DQN

When evaluating the target value, the standard DQN algorithm uses the same values to select and evaluate an action. This increases the chances of selecting overestimated values and therefore having overoptimistic value estimations, called maximisation bias.

To reduce this maximisation bias, Double Deep Q-Network (DDQN) uses two sets of different weights to select and evaluate an action. Weights θ and θ' are obtained by learning two value functions. Equation 2.13 illustrates how the target value is obtained in a DDQN algorithm [49].

$$Y_t^{DoubleQ} \equiv R_{t+1} + \gamma Q(S_{t+1}, \arg\max_a Q(S_{t+1}, a; \theta_t); \theta_t') \tag{2.13}$$

Comparing Equations 2.12 and 2.13 one notices that the action selection is still based on the online weights θ_t but θ_t' is used to evaluate the value of the current policy. The weights can be updated symmetrically by interchanging the role of θ_t and θ_t' while updating the main network [49].

2.5.2 Policy Gradient Methods

Unlike value-based methods which make use of an action-value function to choose an action based on its estimated value, policy optimization methods learn a parameterized policy that selects an action directly for a given state [32]. This allows for the usage of continuous action spaces since the agent can learn a parameterised policy, $\pi(a|s,\theta) \in \mathbb{R}$, to maximise the cumulative reward. It is unfeasible to calculate an action-value estimate for each possible action in each possible state, due to their continuous nature. Furthermore, the policy may be simpler to approximate than the action-value function, leading to faster optimal convergence. Despite this, policy gradient methods suffer from converging to a local optimum and high variance [20].

$$\pi(a|s,\theta) = Pr\{A_t = a|S_t = s, \theta_t = \theta\} \tag{2.14}$$

The policy of a policy gradient method, defined in Equation 2.14, is the probability that an action, a, at time, t, is chosen, given that the environment is in state, s, at time, t, and has policy parameter vector, θ [32].

The optimisation of the policy parameter vector, θ, is based on the maximisation of a performance measure, $J(\theta)$, using gradient ascent, as defined in Equation 2.15

$$\theta_{t+1} = \theta_t + \alpha\widehat{\nabla J(\theta_t)} \tag{2.15}$$

where α is the learning rate and $\alpha\widehat{\nabla J(\theta_t)}$ is the approximate gradient of the performance measure, J, with respect to the parameter vector, θ_t [32].

The performance metric can be defined as

$$J(\theta) \doteq v_{\pi_\theta}(s_0), \tag{2.16}$$

where s_0 is a starting state and v_{π_θ} is the true value function [32]. This definition of the performance metric includes the distribution of states which is affected by the policy parameter itself. The relationship between the policy and the state distribution is generally unknown and therefore, the policy gradient theorem is used to calculate the performance metric irrespective of the state distribution, as defined in Equation 2.17.

$$\nabla J(\theta) \propto \sum_s \mu(s) \sum_a q_\pi(s,a) \nabla \pi(a|s,\theta) \tag{2.17}$$

where $\mu(s)$ is the on-policy state distribution and $\propto$ is the constant of proportionality, which is the average length of an episode [32].

Proximal Policy Optimisation

TRPO was proposed as an improvement to natural policy gradient methods, offering efficient optimisation of large nonlinear policies, such as neural networks. TRPO infers a constraint on the KL divergence, which is the measure of the distance between the old and new policy. This constraint, known as a trust region constraint, allows an increase in the step size, which improves the performance of the gradient ascent [39].

The PPO algorithm [40] builds on top of the TRPO algorithm, being simpler to implement, more general, and having improved sample complexity. There are two variants of the PPO algorithm: adaptive KL Penalty Coefficient PPO, referred to as PPO-Penalty, and clipped surrogate objective PPO, referred to as PPO-Clip.

PPO-Penalty is very similar to TRPO in that it adjusts the KL-divergence at each policy update to meet a target value d_{targ}, where the performance metric is defined as,

$$J^{KLPEN}(\theta) = \hat{E}_t\left[\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)}\hat{A}_t - \beta KL[\pi_{\theta_{old}}(\cdot|s_t), \pi_\theta(\cdot|s_t)]\right] \tag{2.18}$$

where, $\hat{E}_t$ is the empirical average over samples and $\hat{A}_t$ is an estimator of the advantage function [40]. β is computed based on the value of d where,

$$d = \hat{E}_t[KL[\pi_{\theta_{old}}(\cdot|s_t), \pi_\theta(\cdot|s_t)]] \tag{2.19}$$

and

$$\beta \leftarrow \beta/2 \text{ if } d < d_{targ}/1.5 \tag{2.20}$$

$$\beta \leftarrow \beta \times 2 \text{ if } d < d_{targ} \times 1.5 \tag{2.21}$$

On the other hand, PPO-Clip clips the objective function, choosing the lower bound, to remain close to the old policy. Equation 2.22 defines the performance metric of the PPO-Clip algorithm.

$$J^{CLIP}(\theta) = \hat{E}_t\Big[\min(r_t(\theta))\hat{A}_t, clip(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t)\Big] \tag{2.22}$$

where ϵ is a hyperparameter which varies the acceptable interval of $r_t(\theta)$. Obtaining the lower objective ensures that when the objective is to worsen, the probability ratio is used and vice-versa when the objective is to improve.

The PPO-Clip variant was found to achieve superior results when compared to the PPO-Penalty algorithm and therefore, this variant of the PPO algorithm will be utilised in our work [40].

2.5.3 Actor-Critic

As discussed, q-learning based methods learn an action-value function, while policy gradient methods learn a policy function. Actor-Critic methods aim to capitalise on each method's advantages, reducing variance and accelerating learning. In these methods, the critic updates the action-value function while the actor updates the policy parameters in the direction advised by the critic [20]. Deep Deterministic Policy Gradient (DDPG) [50], A3C [38], and SAC [42] are implementations of such methods. A3C is sample inefficient, while DDPG is sensitive to hyperparameter tuning [42]. SAC offers faster and more stable convergence and therefore will be used throughout this work [42].

Soft Actor-Critic

SAC aims to maximise the expected reward while maximising entropy, 'succeeding at the task while acting as randomly as possible' [42]. SAC offers improved stability while achieving state-of-the-art performance on continuous control tasks.

SAC builds on the standard maximisation of the expected sum of rewards, as defined in Equation 2.6, by including the expected entropy of the policy, defined as:

$$J(\theta) = \sum_{t=0}^{T} E_{(s_t,a_t)\sim\rho_\pi}[r(s_t,a_t) + \alpha H(\pi(\cdot|s_t))] \tag{2.23}$$

where α determines the stochasticity of the optimal policy by varying the weighting of the entropy term compared to the reward [42].

This adapted soft Q-value function requires a modified Bellman backup operator, T^π, defined as,

$$T^\pi Q(s_t,a_t) = r(s_t,a_t) + \gamma E_{s_{t+1}\sim\rho}[V_\psi(s_{t+1})] \tag{2.24}$$

where,

$$V(s_t) = E_{a_t\sim\pi}[Q(s_t,a_t) - log\pi(a_t|s_t)] \tag{2.25}$$

is the soft state value function [42].

Being an actor-critic algorithm, SAC defines a soft Q-function, $Q_\pi(s_t,a_t)$, the critic, and a state-value function, $V_\psi(s_t)$, the actor, to work towards a policy, $\pi_\phi(a_t,s_t)$.

The soft Q-function parameters are updated using,

$$\hat{Q}(s_t,a_t) = r(s_t,a_t) + \gamma E_{s_{t+1}\sim\rho}[V_\psi(s_{t+1})] \tag{2.26}$$

To obtain a lower variance estimator, the policy function is re-parameterised using a neural network transformation defined as:

$$a_t = f_\phi(\epsilon_t; s_t) \tag{2.27}$$

where ϵ_t is a noise vector sampled from a fixed distribution. The optimal policy network parameters can now be learnt using,

$$J_\pi(\phi) = E_{s_t \sim D, \epsilon_t \sim N}[log\pi_\phi(f_\phi(\epsilon_t; s_t)|s_t) - Q_0(s_t, f_\phi(\epsilon_t; s_t))] \tag{2.28}$$

The SAC algorithm can also be configured to make use of two Q-functions reducing, the positive bias in the policy improvement step and therefore decreasing training time.

Model-free deep RL methods suffer from two main problems. They require a large number of samples while also being highly sensitive to the hyperparameters chosen. SAC aims to solve these issues, offering better performance, stability and sample efficiency than both on-policy algorithms, with the ability to be applied to high-dimensional tasks. In this work, the twin q-function variant will be utilised.

3 Literature Review

This chapter will review previous work in the field of autonomous driving. Due to the relatively sparse research dealing with the implementation of RL algorithms for the motion planning of tractor-trailer vehicles, this review will first analyse RL implementations for light passenger vehicles navigating through roundabout intersections. Furthermore, rule-based approaches for both light and articulated heavy goods vehicles will be discussed.

Autonomous vehicle navigation is made up of four basic functions: localisation, perception, planning, and control [9]. Some research models a combination of all of these functions, developing end-to-end systems, while others focus solely on a specific function. Both types of systems will be discussed in this chapter.

3.1 Light Passenger Vehicles

Previous research relating to autonomous driving mainly focuses on light passenger vehicles due to their popularity and simpler dynamics model. A vehicle has two main control functions, lateral control, steering, and longitudinal control, acceleration. Different research controls different aspects of the vehicle. While some may only focus on controlling solely the lateral or longitudinal movement, others opt to control both movements. This review will cover both approaches.

3.1.1 Lateral Control Models

Controlling both the longitudinal and lateral behaviour of the vehicle leads to smoother manoeuvres. Despite this, in favour of reducing complexity, some algorithms focus only on the lateral control of the vehicle.

Wolf et al. [26] develop a lane-keeping system by using a visual input to train a DQN algorithm. The Robot Operating System (ROS) is used to combine the track and physics simulation and the algorithm implementation. A discrete action space with 5 different steering angles and a constant velocity was used and two reward systems were tested. They first propose a distance-based reward, affected by the distance to the centre of the lane. To reduce jittering, an action-based extension was used, which negatively rewards counter-steering in turns and a high difference between the vehicle's heading and the lane direction. To favour human-like behaviour, they evaluate their models by measuring their distance to the centre of the lane rather than success percentage. Results show that the action-based extension allows the system to mimic human-like behaviour.

A path-tracking algorithm which combines the robustness of a pure pursuit algorithm and the accuracy and adaptability of RL is proposed by Chen and Chan [16]. The authors note that rule-based approaches require significant fine-tuning based on the application, have high computational complexity, and make use of a simple linear control approach for vehicle driving which is known to be a time-varying nonlinear system. On the other hand, DRL approaches can be considered as black box approaches, which can lead to erratic behaviour where the output control may be unexplainable. Aiming to benefit from the advantages of both systems, a pure-pursuit controller is used to provide a solid starting base, which the RL algorithm fine-tunes to increase tracking accuracy. A continuous state space composed of the target angle and target speed alongside a continuous action space controlling the steering angle are used. The velocity is maintained constant by a separate controller, with the authors noting that including a suitable velocity controller would lead to smoother turning manoeuvres. A reward function based on the lateral position error and the angular error to the centre of the lane are utilised. Weight vectors are used to significantly increase the importance of the lateral error reward when compared to the angular error reward. A cosine wave is used to produce routes with varying curvature to test the limitations of each model. Utilising PPO for its learning efficiency and ability to train using continuous action spaces, the authors store and reuse optimal neural network parameters to improve training performance and stability. Comparing their developed RL and pure-pursuit algorithm to a model solely using the pure-pursuit algorithm, the former can produce low lateral errors at different vehicle speeds and on different routes, even when the pure-pursuit hyperparameters are not optimised. Comparing their developed model to a model predictive control approach on a more difficult route, only the model predictive control approach manages to follow the route. This comes at the cost of significantly higher computational complexity.

3.1.2 Longitudinal Control Models

In order to reduce the problem's complexity and guarantee accurate steering, some research uses a pure-pursuit, path tracking, algorithm to handle lateral control. The RL algorithm is trained to control the acceleration and braking of the vehicle.

Timely roundabout insertion has been researched by Capasso, Bacchiani, and Molinari [12]. Using a modified A3C algorithm with visual inputs, the authors achieve a high success rate with minimal crashes. However, they suffered from a significant performance decrease when testing on unknown roundabouts, which was attributed to the fact that only a few roundabouts were used for training.

Liu et al. [19] integrate RL and imitation learning for a learning-based motion control strategy to navigate through a roundabout intersection which includes traffic. An RGB bird-eye view enriched with map and vehicle information obtained from the

CARLA simulator makes up the state space. Throttle and brake actions are used. Using a reward system based on the velocity, collisions, and proximity to other vehicles, the authors compare several different algorithms. Based on success rate, collision rate, episode reward, and episode length, their combined algorithm using the SAC algorithm and imitation learning, produced superior results to separate baseline algorithms. They also observe that off-policy algorithms achieve superior performance to on-policy algorithms.

Zhang et al. [23] develop a modified actor-critic based RL algorithm for navigating through a multi-lane roundabout with surrounding traffic. Their continuous state space is made up of two components. The first component provides the relative velocity and time headway of the surrounding 12 vehicles while the second component describes the current curvature of the path and the relative angle to the exit. Their discrete action space deals with longitudinal control and a high-level lane change indicator, while it also specifies the aggressiveness of the action performed. Three reward components are utilised. A safety reward based on the distance to the surrounding vehicles. A task reward that determines the correctness of an action by observing its implications. An execution reward to reward lane changes. Testing on known and unknown roundabouts, the authors conclude that the algorithm has suitable decision performance while also being robust enough to work on unknown roundabouts.

Similarly, Wang, Liu, and Wu [27] combine an interval prediction model, which predicts other vehicles' intentions, and a self-attention network to only focus on relevant vehicles. A high-level action space with acceleration, deceleration, and change to the left or right lane actions, is used along with the SAC algorithm. The position, heading, and speed information of the ego vehicle and other vehicles are used as input observations. The ego vehicle is the vehicle which the agent controls and contains the sensors that perceive the environment. Testing on one of the two roundabouts available in the CARLA [51] simulator, the developed algorithm outperforms the standard SAC algorithm with increased stability in velocity, higher rewards, and higher success rates.

Cai et al. [52] develop a LIDAR-based end-to-end autonomous driving model. LIDAR data obtained from the CARLA simulator was converted into an occupancy grid map and combined with map information, such as vehicle routes. The authors aim to maintain temporal dependency and allow the model to predict occlusions, sensor noise, and intents of other vehicles by using three occupancy grid maps. These occupancy grid maps are obtained at 3 different time intervals, 0, 0.5, and 1 second before the current timestep. Steering actions were controlled by a pure-pursuit algorithm, while the vehicle speed was defined by the RL algorithm. A simple reward structure of a positive speed component and a negative collision component was used. The speed component was defined by $v/30$, where v is the velocity of the vehicle while the collision component was a constant value of -50. Training was performed in a crossroad and T-junction with two

different paths for each junction, using the CARLA simulator. Further resembling the real world, occlusions such as smaller vehicles hidden behind larger vehicles were present in training. They use contrastive learning to learn representations of high-dimensional data while also improving sample efficiency. Using a DDDQN algorithm, superior results both in success rate and time for completion were achieved when compared to baseline methods. Such baseline methods included systems using vision, LIDAR, RL, IL, and rule-based approaches. Rule-based approaches were observed to generate over-conservative models with a high success rate at the cost of completion time. Cai et al.'s system achieved success rates comparable to those obtained by rule-based approaches with significantly less completion time. In regular traffic, their model achieved a 93% success rate combined with a 7.6s completion time, when averaging over the 4 intersections tested. On the same intersections and traffic conditions, the TTC based model achieved a success rate of 95% with a 9.4s completion time. Their models also achieved better success rates when testing on a roundabout intersection which was unknown to the algorithms, especially when compared to IL baselines.

In a roundabout scenario, acceleration and high-level lane change actions are determined by combining an RL and rule-based policy. The RL policy is used when the rule-based approach has low confidence [53]. To improve scalability, Cao et al. converted the roundabout into a straight road with multiple on/off ramps. The relative location and velocity of the ego vehicle to the other vehicles around it are used as input to the RL algorithm. The authors make use of a sparse reward function, which only applies a reward of -1 when a collision is detected. This is likely to increase the difficulty of training, but using a simple utility reward may lead to more human-like behaviour. Combining the Intelligent Driver Model (IDM) and Minimizing Overall Braking Induced by Lane Changes (MOBIL) model, as the rule-based algorithms, and a deep q-learning model, they observe that the RL policy learns the weaknesses of the rule-based approach and outputs superior actions. This allows the developed system to achieve better performance than any of the two policies separately.

3.1.3 Longitudinal and Lateral Control Models

Incorporation of both longitudinal and lateral control has allowed for the development of quasi-end-to-end systems [13, 24]. Training the RL algorithm for both longitudinal and lateral control enhances the applicability of the developed models to real-life scenarios, where the acceleration and steering need to be controlled in harmony and not by separate agents whose goals may not be aligned.

Rastelli and Penas [54] tackle the navigation of an autonomous vehicle through a roundabout using fuzzy logic. They developed two fuzzy controllers to follow a generated path. A steering position controller is affected by the angular error and lateral

error. An angular speed controller is influenced by the distance to the bend and the longitudinal speed. Real-life experiments show that the system is stable even when testing at higher speeds and when performing lane changes.

A multi-agent RL approach with the SAC algorithm is applied to navigate through a roundabout with dense traffic. Konstantinidis et al. [13] use the Frenet coordinate system to obtain a relationship between the vehicle's and road's location. 13 observations describe the state of the ego vehicle, its relation with the vehicle ahead of it, and its relation with any surrounding vehicles. These include the current road curvature, inverse TTC to the preceding vehicle, and distance to the closest yield line. The ego vehicle's longitudinal and lateral movements are controlled through a continuous action space. A linear reward function is used where each reward component has a weight relative to its importance. Each reward value is also clipped between a minimum and a maximum value and normalised between 0 and 1. Greater velocity is rewarded to ensure a shorter episode length, while lower longitudinal and lateral acceleration values are favoured for greater driver comfort. A lower lateral offset from the centre of the lane is positively rewarded, while a larger distance to the preceding vehicle is preferred. Any collisions terminate the episode and a relatively higher negative reward is given. Experiments varied in terms of the number of vehicles, their positions, routes, and the amount of noise in the acceleration of other vehicles. The trained model resulted in a larger number of collisions when compared to the real world, despite having a higher-than-average success rate. The algorithm was also evaluated on a different roundabout, resulting in a lower success rate due to a higher number of collisions and timeouts.

Garcia Guenca et al. [24] make use of a Q-learning algorithm to navigate through a multi-lane roundabout. Their system controls both longitudinal and lateral movement, along with handbrake activation. A reward system based on the deviation of the vehicle from the centre of the lane is used. The algorithm utilised the vehicle location and velocity, the radius of the roundabout, and lane information obtained from the CARLA simulator. Experiments were performed with and without traffic and using different exits on a four-exit roundabout. The authors evaluate the developed model by their average speed, reward, deviation from the centre of the lane and distance travelled. Compared to other machine learning algorithms, the authors note that the Q-learning algorithm has improved directionality and a more realistic average speed. Garcia Guenca et al. suggest further exploring roundabouts with different shapes and number of exits to test the robustness of such algorithms.

Chen et al. [55] develop an end-to-end system for urban autonomous driving. Map and route information, and historical vehicle states are combined into a bird-eye view image. This image is processed using a variational auto encoder to learn a low dimensional latent representation, allowing for faster convergence and improved success rates. A reward system based on velocity, steering angle, and collisions is used.

To further improve convergence, the authors use different exploration strategies and frame skip. In frame skip, the same action is used for a number of frames reducing the training complexity. Comparing to DDQN, Twin Delayed DDPG (TD3), and DDPG, their SAC-based system achieved higher success rates alongside faster convergence. When evaluating with traffic, the proposed system has a 58% success rate in reaching the final goal.

Jiang et al. [17] propose an RL-based path-tracking algorithm. The state space is made up of 9 continuous elements including the deviation from the centre of the lane, the angle to the centre of the lane, ego vehicle information, and the velocity of the leading vehicle. The authors make use of a reward function which encourages the agent to reduce the lateral error from the centre of the lane, while also maintaining a predefined velocity. In order to reduce training time, strict constraints are defined for episode termination, which depends on the lateral error, velocity of the ego vehicle, and distance to the leading vehicle. Using a continuous action space, they control the steering angle and acceleration of the vehicle. The authors define two testing scenarios, one with ideal road conditions and another with extreme road conditions. The ideal road is a circular section of road, while the extreme road is a clothoid-shaped road. A clothoid is a curve whose curvature varies linearly based on its length. Such two different road conditions were tested in order to understand the robustness of their RL-based approach when compared to a rule-based approach which makes use of the two PID controllers. Experimental results conclude that on the circular road, both models obtained similar velocity and lateral error values. Despite this, the PID controller obtains higher velocity and lateral errors, when comparing the results obtained on the clothoid-shaped route. While the PID controller obtained a velocity error of 0.37 m/s and a lateral error of -0.08m, the RL controller obtained a velocity error of 0.01 m/s and a lateral error of -0.01m. The authors conclude that the RL controller has higher control accuracy alongside the ability to adapt to changing road conditions.

3.2 Articulated Heavy Goods Vehicles

Due to the physical characteristics such as length, height, and weight of heavy goods vehicles, such vehicles are represented using more complex physical models. Further to this, articulated heavy-goods vehicles further increase this complexity by having a pivot joint connecting the tractor and trailer units. Due to the limited number of RL-based approaches for tractor-trailer vehicles, research using rule-based methods for motion planning of articulated heavy goods vehicles will be discussed.

3.2.1 Rule-based Approaches

Widyotriatmo, Siregar and Nazaruddin [56] develop a line-following control system using the Lyapunov method, where both the velocity and steering angle are chosen. The distance and orientation error from the predetermined path and the angle between the truck and trailer are used as feedback signals. The system is tested by following the perimeter of a square where the algorithm achieves stability, reducing all error metrics to 0.

Diestra and Skouras [14] develop a number of trajectory planners for tractor-trailer vehicles using model predictive control. They first obtain various models of the tractor-trailer dynamics system, which vary in their accuracy and complexity, noting the importance of a reliable model to obtain suitable solutions. Using the Euler-Lagrange equation, they discretise the continuous dynamic system of the tractor-trailer vehicle into three different models. In favour of lower complexity, allowing for the developed models to be realistically usable, no model can represent the tractor-trailer system in all states. The authors also note that the 3 models have different conditions under which they provide accurate representations, such as low or high velocities. This is due to the assumptions made to have simpler models, such as neglecting the mass and inertia of the system. Simulating their work using ForcePro Solver, the authors tested their trajectory planners on two tight right-hand turns and a roundabout scenario where the vehicle navigates through 270° before exiting. All models produce low estimation errors where more complex models achieve superior results at the cost of greater solving time.

Using sequential quadratic programming, Oliveira et al. [15] develop a model for buses and tractor-trailer vehicles to optimally navigate roundabouts and s-turn intersections. Optimal driving behaviour is defined as the swept area by the vehicle being centred on the road. The swept area is the total area covered by the vehicle while navigating through an intersection. This implies that the objective of the model is to maintain the far-left and far-right parts of the tractor-trailer vehicle equidistant from the road boundary. The authors note that for this to occur, the rear wheels of the tractor and trailer units may not be at the centre of the road. Testing in a roundabout scenario, the tractor-trailer vehicle is observed to maintain equal distances to both sides of the road boundary. A slight deviation occurred while entering and exiting the roundabout where the planner achieved a maximum and minimum width of 2.3m and 2.26m, while the ideal width was 2.27m. They also considered routes with varying curvature, resembling real-life scenarios, where the developed model was also successful.

3.2.2 Machine Learning based Approaches

Zhang, Eck, and Lotz [57] develop a path-planning approach for tractor-trailer vehicles using semi-supervised learning. A Gated Recurrent Unit (GRU) based encoder-decoder deep neural network is used to generate suitable paths. These sets of waypoints are given a score using a path cost function which is used to learn which paths minimise the low-speed swept area of the tractor and trailer. The path cost function is mainly based on the tractor's and trailer's deviation from the standard path, the tractor's yaw acceleration, and collision avoidance. The authors use an intermediate and vector representation to represent the obstacles and lane features. Random variations of a fifth polynomial arc and an arc preceded by a line segment, are used as training lanes in which, at maximum, two obstacles are generated along the route. For routes with no obstacles, the authors note that the model learned to stay within the lane's borders. When faced with a lane containing obstacles, a new path is successfully generated to avoid collisions in 99.33% of the cases.

3.2.3 Reinforcement Learning Approaches

Olcay, Rui, and Wang [30] compare several variants of the DQN algorithm and SAC algorithm for controlling the acceleration and steering of a farm tractor combined with a trailer to perform a 360° turn, known as a headland turn. Their input state consists of 5 distance sensors, covering the front and sides of the tractor, which obtain the distance to the closest obstacle. The authors utilise a reward system split into three phases. When the agent is alive, it positively rewards the agent with a ratio of the distance travelled and negatively rewards the agent based on the distance to the next waypoint. When the agent collides with the road boundary or successfully completes the task, a positive reward based on the number of checkpoints passed and the number of timesteps completed is awarded. When a collision occurs, a negative reward for the collision is given. A discrete action space composed of three possible actions is utilised. When observing the training results, more complex variants of the DQN algorithm, such as the DDQN and DQN with prioritized experience replay, obtained significantly faster convergence rates than the standard DQN algorithm. Moreover, the SAC algorithm obtained even faster convergence rates than any of the DQN variants. Using a pixelated 2D testing environment, the SAC algorithm obtains the shortest episode times when compared to all DQN variants.

DQN and DDDQN algorithms were compared by Serin and Soodla [25] for the control of a semi-truck and trailer combination navigating through a roundabout and an uncontrolled T-junction. Two types of action spaces were tested. A speed-based action space, which sets the velocity of the ego vehicle, and an acceleration-based action

space, which alters the acceleration of the vehicle, mimicking human-like control. Only longitudinal actions were considered since the vehicle followed a pre-planned route. Simulation of Urban MObility (SUMO) [58] was used as the simulation environment and compared to DQN, DDDQN produced safer results while taking longer to complete the task. Despite this, these algorithms solve the scenarios much faster than TTC, obtaining similar success rates, at the cost of having a higher number of timeout states.

Ren et al. [59] develop a double DQN based path tracking algorithm for an agricultural tracked tractor and trailer vehicle. Such trailer has a length of 3.6m. The authors note that vehicles which use tracks instead of wheels have different dynamic properties and therefore models cannot be interchanged between both vehicles. Aiming to maintain the trailer at a constant distance away from a predetermined path, the authors utilised 3 actions: straight, turn left, and turn right. Radar was used to obtain a map illustrating the boundaries of the path. This map was centred at the midpoint between the tractor and trailer and was converted into 360 points to be processed as input to the algorithm. A sparse reward system was developed around the expected action to be taken by the agent, depending on the trailer unit's position. A reward of 0 is given when the lateral deviation is 0. Depending on the heading and lateral deviation of the vehicle, a reward of +1 is given to the agent if the correct action is chosen and -1 if the incorrect action is chosen. A simple training path consisting of a straight section, 45° right turn, a straight section, 45° left turn, and another straight section was used. The generated model was tested in a simulation environment, which consisted of a U-shaped path, with the circular part of the path having a diameter of 4m. The generated models were tested at two different speeds, 1.4km/h and 2.7km/h. When operating at the higher velocity, an average lateral deviation of 0.040m was observed, while when operating at 1.4km/h an average lateral deviation of 0.038m was calculated. The authors also performed a real-life field test, where a higher lateral deviation of 0.074m and 0.078m was observed when operating at 1.4km/h and 2.7km/h respectively.

Wang et al. [28] make use of a DDQN algorithm to generate a lane-changing model for a semi-truck and trailer vehicle. They develop a high-fidelity truck and trailer model with realistic lateral and longitudinal kinematics, but at the time of writing, such a model is not publicly available. A predictive cruise control algorithm is used to maintain the vehicle's longitudinal control, while an action space consisting of change to the left lane, change to the right lane, or stay in the current lane, is used to laterally control the vehicle. The vehicle's location, velocity, and position of neighbouring vehicles are observed from the environment. The last four sets of observations are used as input to the algorithm. A reward component that penalises lane changing is used. Using a combination of TruckSim[1] for the vehicle kinematics, SUMO for the traffic simulation and CARLA for the driving simulation, they train and evaluate the model with three traffic densities

[1] https://www.carsim.com/products/trucksim/index.php

and two vehicle speeds. The generated models produce superior results compared to rule-based models, by avoiding multiple lane changes and therefore reducing the risk of an accident. Such vehicles have a higher risk of accidents when compared to smaller vehicles due to their physical characteristics.

3.3 Summary

In this chapter, a detailed review of recent literature related to autonomous navigation of vehicles around roundabout intersections was performed. Due to the relatively sparse research related to articulated heavy goods vehicles, the initial part of the review focused on different methods used for light passenger vehicles. Review related to articulated heavy goods vehicles consisted of rule-based and RL-based methods.

In a bid to reduce computational complexity, some research opted to only control the lateral movement [16, 26], by using steering actions, or longitudinal movement [23, 52], by using acceleration actions. The movement not controlled by the RL model was usually managed by a rule-based approach, such as a pure-pursuit algorithm or a PID. Other approaches aiming at obtaining quasi-end-to-end models, controlled both movements using the RL algorithm [24, 55].

The input observations vary from values related to the state of the vehicle [17, 27] to more complex observations, such as LIDAR [52] and images [55]. All types of inputs were successfully used to generate usable models.

Reward functions also varied significantly. Some approaches meticulously define the different states the agent can find itself in and the reward associated with being in that state [17, 55]. Some examples of such rewards include a reward based on the distance of the vehicle to the centre of the lane and a reward based on the velocity and distance to the leading vehicle. Such reward functions make use of different techniques, such as weighting, clipping, and normalisation, to obtain the optimal ratio between each reward component. Other approaches significantly simplify the reward function by only rewarding positively proportional to the vehicle speed and negatively for collisions [52].

The algorithms utilised varied from on-policy algorithms, such as PPO [26], to off-policy algorithms, such as DQN [30] and DDDQN [25]. Others also utilised actor-critic based methods, such as A3C [12] and SAC [27]. While all models using different types of algorithms managed to obtain suitable results, Liu et al. [19] state that for solving the motion control of a vehicle navigating through a roundabout intersection with traffic, off-policy algorithms obtained superior results when compared to on-policy ones.

Choosing the right evaluation metrics to assess the developed models is key, allowing for suitable comparisons. Episode length and episode reward are commonly used when assessing RL models. Approaches relating to autonomous vehicles also include

success and collision rates, along with the average distance to the centre of the lane. Specific to articulated vehicles, Oliveira et al. [15] aim to obtain a model which minimises the swept vehicle area.

Rule-based research [14, 15] dealing with the path-tracking of tractor-trailer vehicles makes use of mathematical models to represent the physical movement of the tractor-trailer vehicle. As mentioned by Diestra and Skouras [14], such models vary in their complexity. Simpler models reduce the accuracy of the model but enjoy lower computational complexity. On the other hand, complex models are able to represent the vehicle accurately in more scenarios but suffer from high computational complexity. This limits such approaches by, for example, only working at lower speeds. RL-based approaches work by trial-and-error and therefore are able to obtain superior performance in unseen environments.

As one can observe, there is a limited amount of research using RL-based approaches for the motion control of tractor-trailer vehicles. Despite making use of an RL framework for the lateral and longitudinal control of a tracked agricultural tractor-trailer vehicle, Ren et al. [59] note that the tractor used has different physical properties than wheeled vehicles. Furthermore, the trailer is only 3.6m in length. The trailer utilised in this work has a length of 13.6m. Such an increase in length greatly affects the behaviour of the tractor-trailer vehicle, increasing its low-speed swept area, since the trailer's wheels are further away from the articulation joint. Finally, due to being designed for the farming environment, Ren et al.'s [59] framework utilised very low velocities of, at maximum, 2.7km/h. Such low velocities are not appropriate for road vehicles.

While the work of Serin and Soodla [25] only focuses on the longitudinal control of the vehicle and uses a predetermined path to laterally control the vehicle, this work will focus on laterally and longitudinally controlling the vehicle using an RL algorithm. Despite knowing a path going from the starting location to the ending location, this path is only used as a reference. If the tractor-trailer vehicle were to follow this path, the model would not successfully be able to complete the routes, with the trailer unit terminating the episode by colliding with the kerb.

Several input observations utilised in previous research will be employed. These include distance sensors and information related to the tractor-trailer vehicle. As Know et al. [60] discuss, specifying several reward components in hopes of improving the behaviour of the agent, known as reward shaping, may lead to decreased overall performance. The agent may exploit the reward system, resulting in sub-optimal behaviour. In light of this, our work will focus on reducing the complexity of the reward function, by limiting the reward-shaping functions utilised.

Observing their use and success in previous research, PPO, DQN and SAC algorithms will be tested and compared in this research. Such algorithms use different underlying approaches, where PPO is an on-policy, policy gradient based method, DQN

is an off-policy, q-learning based method, while SAC is an off-policy actor-critic based method. In order to gauge the performance of each model, similar evaluation metrics utilised in previous research will be employed.

At the time of writing, our work, which focuses on the motion control of a tractor-trailer vehicle through a roundabout intersection using lateral and longitudinal movements is the first-of-its-kind.

4 Methodology

This chapter describes the development and implementation of each objective. Implementing an RL algorithm on a tractor-trailer vehicle requires several preliminary steps. After choosing a high-fidelity simulation environment, a tractor-trailer vehicle model, and various roundabout scenarios, need to be developed. Each component is then designed, justifying any decisions taken.

4.1 Objective 1: Simulation Components

4.1.1 Simulation Environment

Several open-source simulation environments have been developed for testing RL algorithms. Simulators used in autonomous driving research include LG Silicon Valley Lab (LGSVL) [61], SUMO [58], and CARLA [51]. LGSVL lacks an active community which hinders its use. SUMO has been used for training the control function of autonomous vehicles. Being designed as a traffic simulator, SUMO lacks realistic vehicle control, advanced physics simulation for vehicle movement, and more complex perception inputs, such as LIDAR.

CARLA is an open-source autonomous vehicle driving simulator, supporting a large number of sensors, including RGB camera, LIDAR, radar, collision, Inertial Measurement Unit (IMU). Traffic with different levels of aggressiveness can also be generated. Various vehicle properties, such as velocity and sensor outputs, can be obtained through the Python Application Programming Interface (API) library, which also enables the control of such vehicles by defining the acceleration and steering angle.

Furthermore, CARLA computes and takes into consideration several physical properties. Due to the size and weight of tractor-trailer vehicles, several physical forces, not present in light passenger vehicles, affect their movement. Firstly, the weight of the trailer can vary significantly, which in turn affects the acceleration and braking performance of the tractor unit. Secondly, the high centre of mass of the trailer can lead to rollover, where the trailer rolls on its side due to its speed and turning angle. Furthermore, CARLA is also able to correctly simulate the articulated point joining the tractor and trailer and how it affects both units. Such a high-fidelity simulation software allows for the generation of quasi-real-life data, giving further confidence that the results obtained can be replicated in real-life scenarios.

CARLA also allows the creation of custom vehicles and maps. Such a feature is essential to this research since, at the time of writing, no autonomous driving simulator offers a ready-made tractor-trailer model and suitable roundabout scenarios.

4.1.2 Development of custom tractor-trailer model

There are several steps in building a custom vehicle in CARLA. The developed tractor and trailer models used in this research are based on the work of Engles[1]. Despite having both the tractor and trailer visual and physics models, improvements to both models were necessary to more accurately mimic real-life behaviour. Using Unreal Engine[2], the simulation development software used by CARLA, vehicle meshes generated using Blender[3], were integrated into the models to allow for correct collision detection, as shown in Figure 4.1.

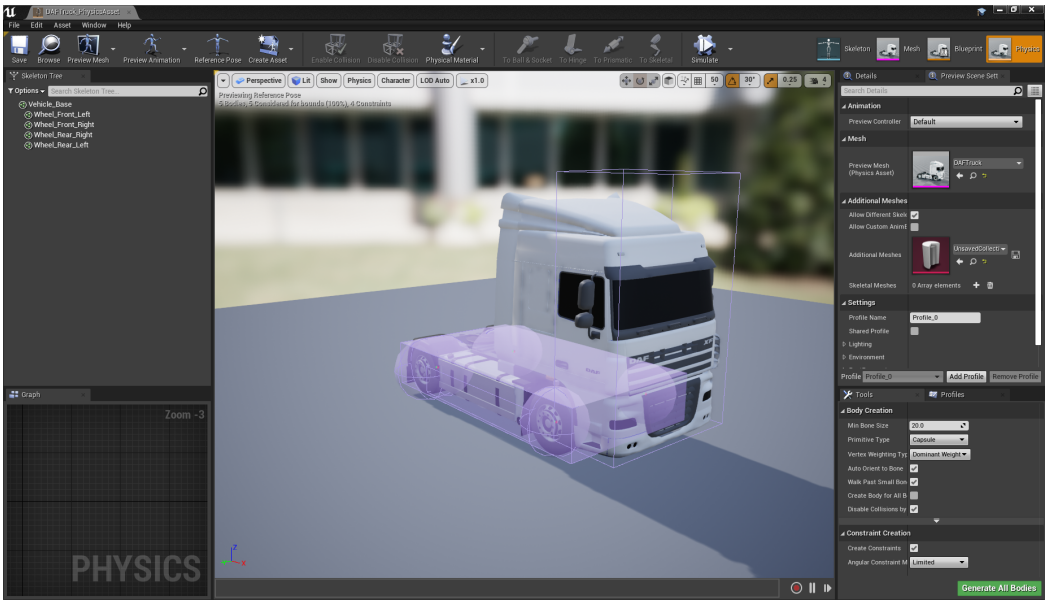


Figure 4.1 Collision meshes for tractor unit.

Known bugs reported by Engles, such as the tractor and trailer units not connecting together through the articulated joint, the tractor occasionally not being able to move forward freely, and the tractor-trailer vehicle not being able to move in reverse, were also fixed.

Figure 4.2 illustrates the completed tractor-trailer vehicle. The physical specifications such as the length, width, height, and weight of both units were verified to be similar to real-life vehicles by referencing similar tractor, the DAF XF 450 FT[4], and trailer[5] vehicle specifications. The specifications used are detailed in Table 4.1 and are compliant with the legal requirements defined by the European Union [62]. Ensuring that the tractor-trailer model highly resembles a real-life vehicle, combined with the high-fidelity

[1] www.github.com/frankeng/CarlaSemiTruckTrailer
[2] www.unrealengine.com
[3] www.blender.org
[4] www.daf.co.uk/en-gb/trucks/specsheets-search-page
[5] www.dsv.com/en/our-solutions/modes-of-transport/road-transport/trailer-sizes/box-trailer

simulator, increases the confidence that the obtained results can be replicated in a real-world environment.

The tractor-trailer vehicle is publicly available on GitHub[6], facilitating future research related to tractor-trailer vehicle.

Figure 4.2 Completed tractor-trailer vehicle.

Table 4.1 Tractor and trailer vehicle specifications

Vehicle	Length (m)	Width (m)	Height (m)	Weight (kg)
Tractor	5.80	2.40	3.80	8000
Trailer	13.60	2.40	3.80	7000

4.1.3 Development of custom scenarios

At the time of writing, CARLA only offers two roundabouts scenarios. One roundabout is a single-lane roundabout with one exit, at the same point of entry, while the other roundabout is a two-lane roundabout with four exits. In order to correctly train and test the developed models, roundabouts with different number and position of exits, and diameters, need to be utilised. RoadRunner [63] is a 3D design tool that is used to create scenes for driving simulation. Such scenes can then imported and used in CARLA.

There are various different types of roundabouts which differ in, the number of exits and their placement, the number of lanes, ranging from one to six lanes, and diameter. After observing multiple roundabout intersections in Malta's road network through Google Earth[7], two-lane roundabouts with varying diameters were noted to be the most common. Most of these roundabouts have between 3 and 4 entry-exit points.

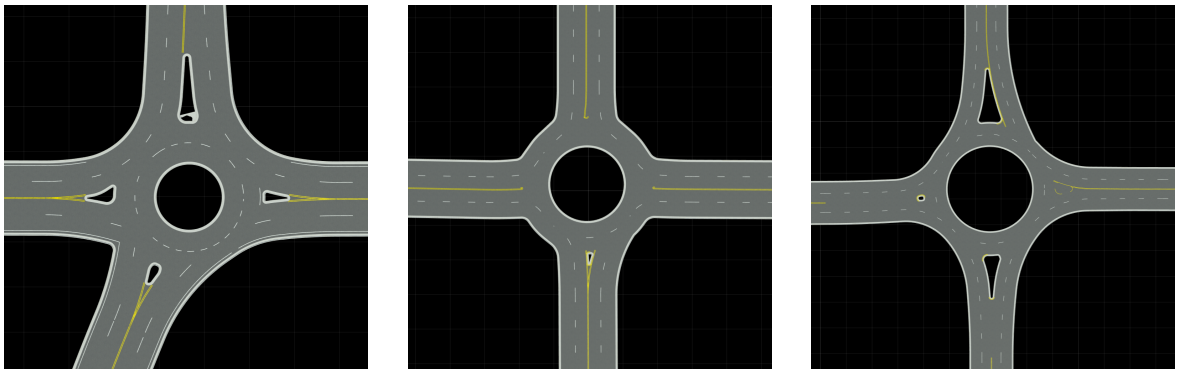
[6] www.github.com/DanielAtt2000/Tractor-Trailer-Vehicle-and-Roundabout-Dataset-Carla
[7] www.earth.google.com

The entry-exit points of such roundabouts significantly differ in the angle at which they approach the roundabout. Therefore, simulating two-lane roundabouts with 3 and 4 entry-exit points and varying diameters provides a suitable representation of real-world intersections.

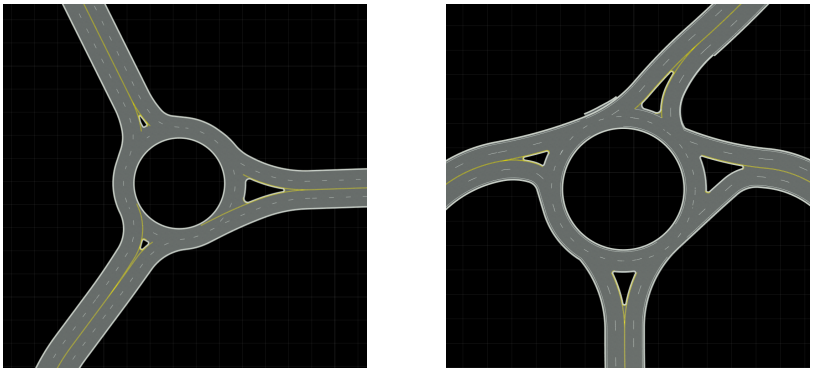
While developing these scenarios, a 16m diameter roundabout was found to be small enough in diameter such that it forces the tractor-trailer to navigate a wider path, crossing into other lanes and deviating from the original path, in order to avoid collisions. This may be necessary for both manoeuvring around the roundabout and for entering or exiting the roundabout. On the other hand, a 50m diameter roundabout was large enough that the tractor-trailer vehicle could follow the centre of the lane while still completing the turn without any collisions, due to the low turning angle. Three more roundabouts with diameters in between these two extreme values were chosen for adequate variation in the training and testing data, improving the generalisability of the model. Two-lane roundabouts were chosen to be developed since the inner and outer lanes offer two paths with slightly different curvatures. This, along with the greater variety of the entry and exit angles allows for more variation in the training and testing examples, leading to more confidence in the obtained results. Two-lane roundabouts also help visualise the unconventional path the tractor-trailer vehicle has to take while navigating tight radius paths by crossing into other lanes.

As illustrated in Figure 4.3, five different roundabouts have been developed using the right-hand traffic system. These differ in their diameter, number and angle of exits. The developed roundabout scenarios have diameters of 16m, 20m, 32m, 40m, and 50m. The 40m roundabout has 3 entry-exit points while the rest have 4 entry-exit points with each roundabout having different placement and angle of entry-exit points.

As can be seen in Figures 4.3a, 4.3b, 4.3e, a number of their entry-exit points are designed with low radius sections for which the tractor-trailer vehicle has to follow an unconventional path. In the 16m diameter roundabout, Figure 4.3a, this can be seen in the route going from the west entry point to the south exit point in the outer lane. In the 20m diameter roundabout, Figure 4.3b, this can be observed at every connection between the roundabout and the entry-exit lanes where if the tractor-trailer vehicle is following the outer lane, the vehicle must deviate so that the rear of the trailer does not collide with the kerb. In the 50m diameter roundabout, Figure 4.3e, the west entry point is composed of a tight turning radius, while this is also true for the east exit lanes. Such road sections require the tractor-trailer vehicle to follow an unconventional path, increasing the data for such instances while also providing examples of cases where the vehicle has to deviate away from the centre of the lane while entering or exiting the roundabout and not only while circulating the roundabout. In comparison, all entry-exit points in the 32m and 40m diameter roundabouts, shown in Figures 4.3c, 4.3d, are developed to have a low curvature path.



(a) 16m diameter (b) 20m diameter (c) 32m diameter



(d) 40m diameter (e) 50m diameter

Figure 4.3 Developed roundabout scenarios having different diameters.

Each roundabout is approached by two, 3.7 meter wide lanes. These connect onto the two-lane roundabouts. This allows for a total of 5 possible routes per entry for 4 exit-entry roundabouts, as shown in Figure 4.4. Similarly, 3 exit-entry roundabouts have a total of 4 possible routes. Each route is defined as a list of waypoints at the centre of the appropriate lane. No modifications were made to the waypoints where tighter radii occur.

After importing both custom assets into CARLA and doing the appropriate modifications, a packaged version of CARLA was generated to be able to have multiple environments running at the same time, allowing multiple concurrent RL training workers.

The 16m, 32m, and 50m roundabouts were used for training the RL model. One route out of the 52 routes available was randomly chosen during each training episode. The 20m and 40m roundabouts were used for testing the RL model. Testing using roundabouts with different physical properties than those used during training gives more confidence in the generalisability of the model. None of the 23 routes used during testing were available to the model when training.

The roundabout intersections are publicly available on GitHub[8]. Such roundabout dataset can be used as a benchmark for future research.

[8] www.github.com/DanielAtt2000/Tractor-Trailer-Vehicle-and-Roundabout-Dataset-Carla

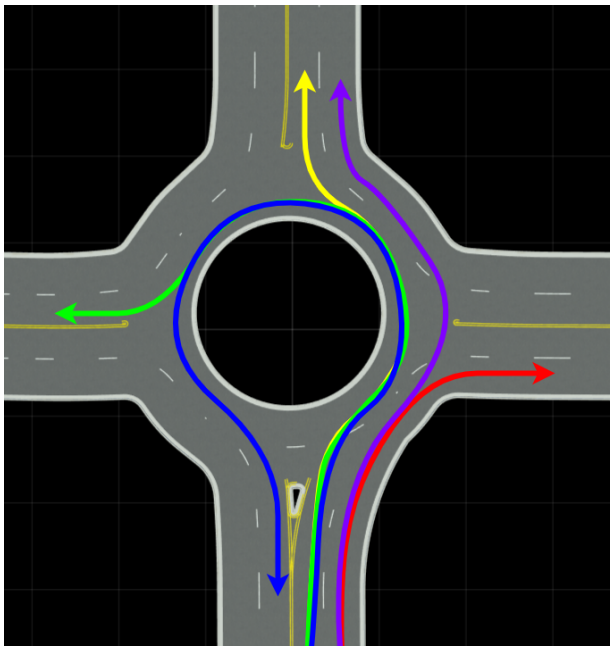


Figure 4.4 The 5 possible routes for 4 entry-exit roundabouts.

4.2 Objectives 2: Reinforcement Learning Environment

4.2.1 System Overview

In order to solve our autonomous driving problem, a combination of different software was required. While the RLlib [64] framework dealt with the training of the RL algorithm, the CARLA simulator [51] was used to simulate the movement of the tractor-trailer vehicle in the roundabout intersections.

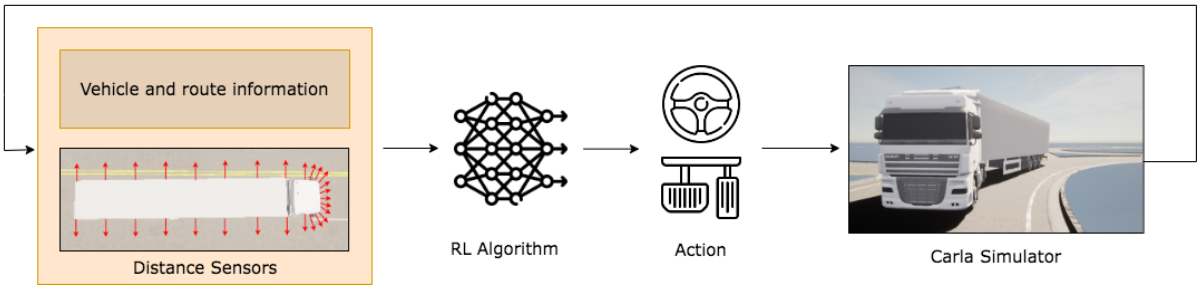


Figure 4.5 An overview of the system architecture.

Figure 4.5 illustrates an overview of the system workflow. This is based on the RL interaction loop illustrated in Figure 2.2 on page 7. The CARLA simulator [51] is used to obtain observations of the current state of the vehicle through the provided Python API. These, along with the reward from the previous state, are fed to the RLlib [64] framework to train the RL algorithm. The RL algorithm then outputs the desired actions which are performed in the simulation environment, progressing to the next set of observations.

The communication between the CARLA simulator and the RLlib framework is facilitated by using the Gymnasium framework [65]. This framework provides a standard set of APIs that are called by the RL framework to obtain observations, execute actions, and reset the environment, among other functions. These functions are then implemented to communicate with the simulator, in our case using the Python API provided by CARLA, to retrieve the observation values, perform actions, or reset the environment. A visual representation of this communication is shown in Figure 4.6.

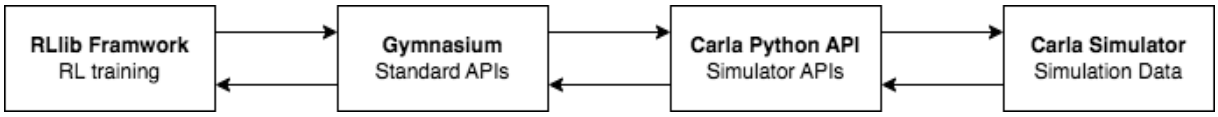


Figure 4.6 Communication between the RLlib framework and the Carla Simulator.

It should be noted that for each episode during training, random starting and ending points are chosen. The location and rotation of the waypoints that compose the route between the starting and ending points are known by the system.

In our work, we utilised CARLA's implementation[9] for the interface between the RLlib framework and the CARLA simulator. Along with other updates and improvements, such implementation was modified to allow for the control and retrieval of sensor data from two vehicles, the tractor and trailer unit.

4.2.2 Reinforcement Learning Environment

Our autonomous driving task is represented by the state space, S, available actions, A, reward function, R, state-transition model, f, and discount factor, γ. The following sections will describe the details of each component, justifying any decisions taken.

State Space

The state space used to describe the tractor-trailer vehicle and the surrounding environment is a one-dimensional vector of 69 continuous elements defined by:

$$S = \langle v, d_c^t, \theta_{tt}, d_r, d_w, d_l, \theta_{tk\_b}, \theta_{tl\_b}, \theta_{tc}, \theta_w, r \rangle \tag{4.1}$$

where:

- v is the velocity of the tractor-trailer vehicle
- d_c^t is the distance between the centre of the tractor unit and the centre of the lane
- θ_{tt} is the angle between the tractor and trailer units

[9] www.github.com/carla-simulator/rllib-integration

- d_r is a vector of 29 distances to the road boundary
- d_w is a vector of 2 hypotenuse distances from the tractor unit to the next two waypoints
- d_l is a vector of 2 distances to a line generated at right angles to the next two waypoints
- θ_{tk_b} is a vector of 5 angles between the tractor's forward vector and the forward vector of future waypoints
- θ_{tl_b} is a vector of 5 angles between the trailer's forward vector and the forward vector of future waypoints
- θ_{tc} is a vector of 5 angles between the tractor unit and the centre of the lane
- θ_w is a vector of 8 angles subtended by two equally distanced waypoints from a central waypoint
- r is a vector of the next 10 radii values along the route

These can be split up into three categories namely: vehicle information, future position and heading, and curvature of the route

Vehicle Information The vehicle reacts differently to the same action depending on its speed and therefore knowing the velocity of the tractor-trailer vehicle, v, the agent can learn this relationship. d_c^t is the distance between the centre of the tractor unit and the centre of the lane. The optimal path a vehicle should follow can be defined as that when the centre of the vehicle follows the centre of the lane. Such an input allows the agent to understand how far away the tractor-trailer vehicle is from following the optimal path. Since the tractor and trailer are connected with an articulation joint, the angle between them, θ_{tt}, is provided to the agent to understand the rotation of the trailer with respect to that of the truck.

In order for the agent to understand its distance away from the road boundary, d_r, the tractor unit is equipped with 13 distance sensors at 15° intervals, originating from its centre, while the trailer has 8 equally spaced sensors on each side. These are illustrated in Figure 4.7. Models using a lower amount of distance sensors were also tested, but these obtained inadequate results. Multiple distance sensors were used at various angles at the front of the tractor unit to ensure that the agent has a clear picture of the obstacles ahead. Due to the length of the trailer, the distance to the road boundary varies significantly along its length. This is especially true on smaller-diameter roundabouts, where the length of the trailer may be equivalent to the diameter of the roundabout. Furthermore, while turning, the section of the trailer in front of the articulation

point deviates in the opposite direction to that of the rear part of the trailer. Therefore, multiple distance sensors spaced throughout the length of the trailer are required for the agent to get an accurate picture of the trailer's position relative to the road boundary.

Since CARLA does not offer a distance sensor, LIDAR sensors were repurposed for calculating such distance by narrowing their horizontal field of view to 2° and taking the distance to the closest point detected. The maximum range of this sensor was set to 7m which was found to be an adequate distance.

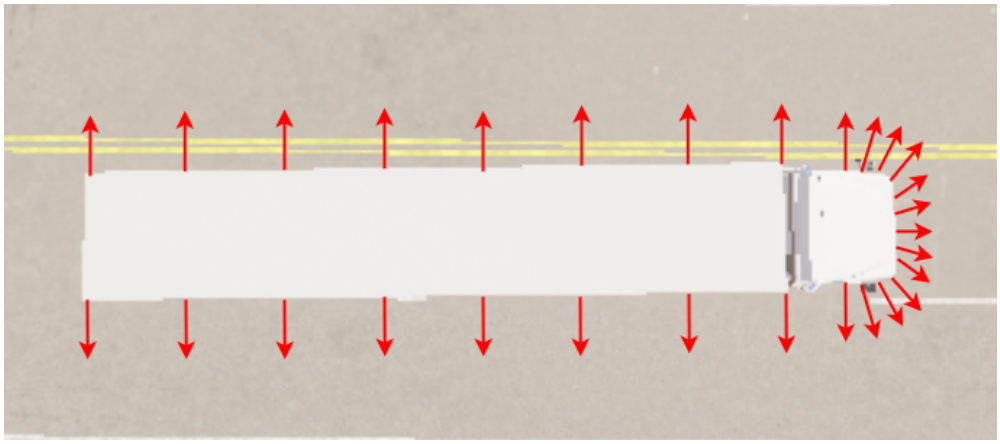
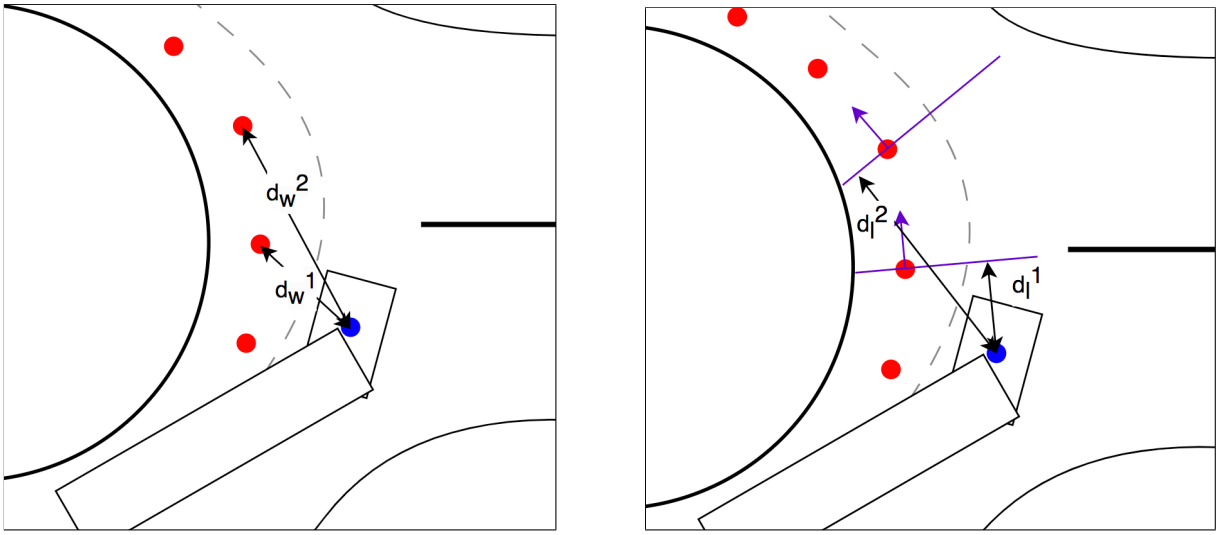


Figure 4.7 The 29 distance sensors found on the tractor-trailer vehicle.

Future Position and Heading The agent is also given information regarding its expected future position and heading so that it can move the vehicle accordingly. It should be noted that, as clarified in Section 4.1.3, the waypoints are at the centre of the lane and no modifications were made to their position or heading to accommodate for the unconventional path the tractor-trailer vehicle may need to take. This implies that the values calculated below make use of the expected future waypoints from which the tractor-trailer vehicle may need to deviate away from. If the tractor-trailer vehicle were to follow the path provided, the trailer would collide with the kerb on most routes.

The hypotenuse distance from the tractor unit to the next two waypoints, d_w, is provided to inform the agent if the vehicle is moving in the right direction. When taking tighter radii turns, the tractor has to move away from the route and therefore a line at right angles to the forward vector of a waypoint, can be used to inform the agent that despite not moving towards the waypoint, the general direction is correct, moving closer to the end of the route. Such a line is generated for the next two waypoints and the closest distances to these lines are calculated, d_l. Figure 4.8 illustrates how these two observations are calculated. In both Figures 4.8a and 4.8b, the red dots are the route waypoints, the blue dot is the centre of the tractor unit and the double-ended arrows are the respective distances. In Figure 4.8b, the purple arrow represents the forward vector of each waypoint while the purple line is at right angles to the forward vector.



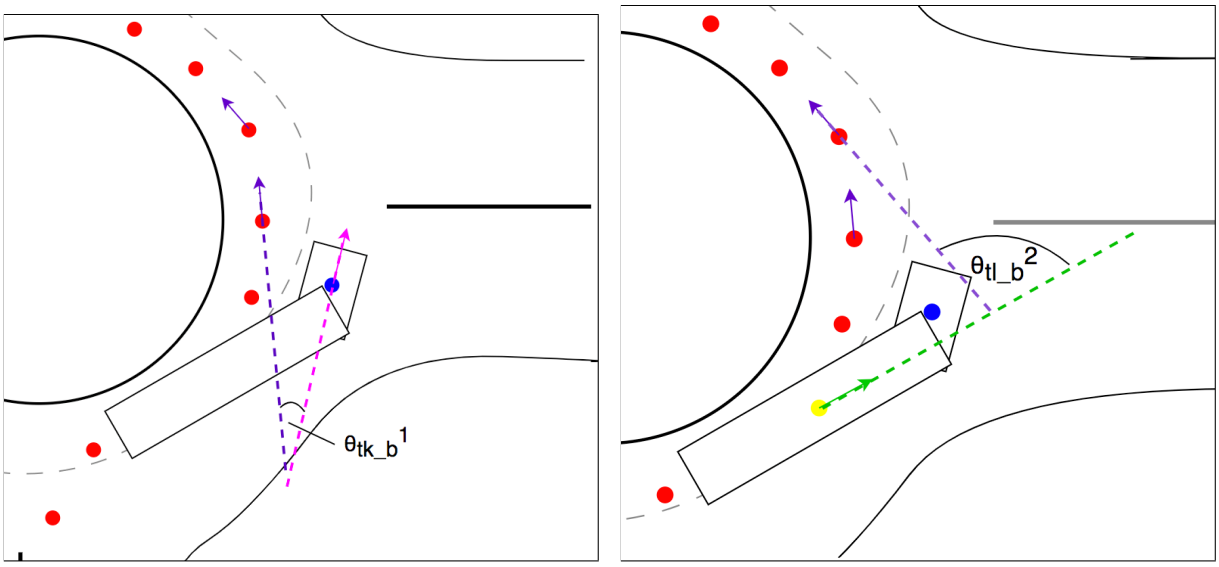
(a) d_w (b) d_l

Figure 4.8 Illustration for calculating observations d_w and d_l for the next two waypoints.

For the agent to comprehend its expected future heading, the angle between the tractor's forward vector and the forward vector of future waypoints is obtained, θ_{tk_b}. This angle is calculated for 5 different waypoints, 1,2,5,7, and 10 waypoints ahead. Due to the articulated joint, the trailer's heading can widely vary from that of the tractor, and therefore these angles are also calculated for the trailer unit, θ_{tl_b}, at the 5 different waypoints. The sign of the angle denotes the direction of the waypoint relative to that of the tractor or trailer. Figure 4.9 illustrates these concepts where the red dots are the route waypoints and the purple arrows indicate their forward vector. In Figure 4.9a, the pink arrow indicates the heading of the tractor unit, while in Figure 4.9b, the yellow dot is the centre of the trailer unit with the green arrow being its forward vector. These forward vectors can be extrapolated up to a point at which they intersect from which the angle between the two forward vectors can be calculated.

The agent is also given the tractor's angle to the centre of the lane. A visual illustration of how this angle is computed is shown in Figure 4.10. Two vectors, one from the previous waypoint, b, to a future waypoint, c, and another from the previous waypoint, b, to the tractor unit, a are determined and the angle between these two vectors is the tractor's angle to the centre of the lane, θ_{tc}. This is calculated for 5 future waypoints, 1,2,5,7, and 10 waypoints ahead.

The above values are calculated using several future waypoints in order for the agent to get a better understanding of the expected future positions. This allows the agent to recognise that, in order to reach future positions, an unconventional path has to be taken. The tractor-trailer vehicle may need to be navigated away from the conventional path from the very start of the route, depending on the curvature of the route, to be able to successfully complete it.



(a) θ_{tk_b} calculated for a waypoint, 1 point ahead

(b) θ_{tl_b} calculated for a waypoint, 2 points ahead

Figure 4.9 Illustrations for calculating observations θ_{tk_b} and θ_{tl_b}.

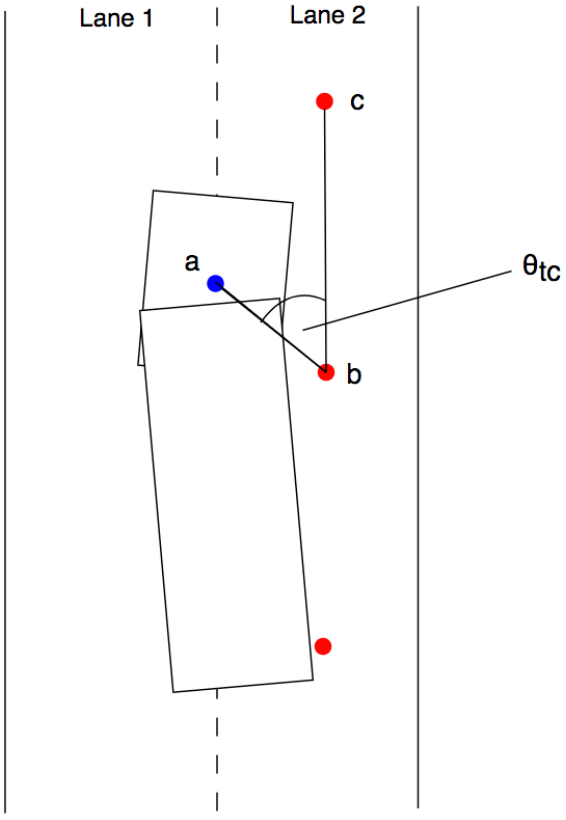
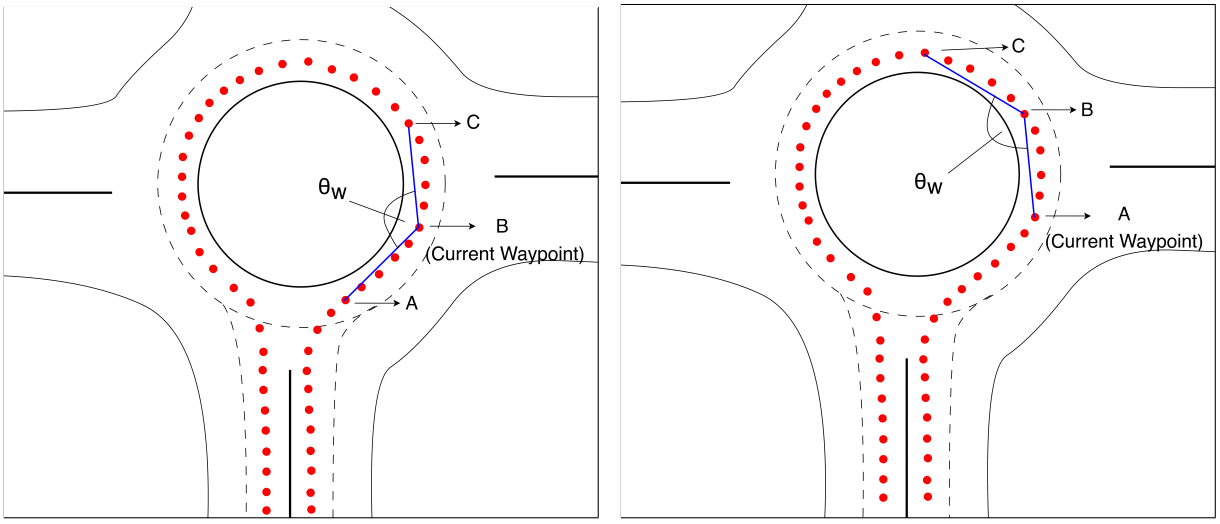


Figure 4.10 Visual illustration for calculating the angle of the tractor unit to the centre of the lane with 1 waypoint ahead, θ_{tc}.

Curvature of the Route To represent the curvature of the route, not the path the tractor-trailer needs to take, the angle subtended by two equally distanced waypoints from a central waypoint is calculated, θ_w. With the current waypoint being used as the central waypoint, 4 angles are calculated where the end waypoints are 5, 7, 10, and 12 waypoints away from the central one. This is calculated as shown in Figure 4.11a, where the angle between $\vec{AB}$ and $\vec{BC}$ is calculated. In Figure 4.11 the case of having a difference of 5 waypoints is being illustrated.



(a) θ_w using the current waypoint as the central waypoint

(b) θ_w using the current waypoint as one of the end waypoints

Figure 4.11 Visual illustration for calculating observation θ_w.

Taking the current waypoint as the central waypoint generates a value which describes the current curvature of the route. On the other hand, using the current waypoint as one of the end waypoints, obtains a value which represents the future curvature of the route. This is illustrated in Figure 4.11b where the angle between $\vec{AB}$ and $\vec{BC}$ is calculated. This is performed for 4 different distances, where the central waypoint is equally spaced, 5, 7, 10, and 12 waypoints away from the end waypoints, and one of these end waypoints is the current waypoint. Both of these values are essential for the agent to understand the current and future curvature of the route and be able to choose the appropriate action.

Further to this, the radius of the route is approximated at 10 different intervals by generating a chord between every 5 future waypoints and calculating their bisecting line. The intersection between two bisecting lines of adjacent chords can be taken as the centre of a circle whose radius can be calculated, r. This is visualised in Figure 4.12 where the blue lines indicate the generated chords. The green, orange, and pink lines are the bisecting lines of each chord. Since the first two chords are almost parallel to each other, their bisecting lines intersect at a distant point and therefore their intersection is marked by the dashed orange and green lines. The centre o_1 can be used to calculate the

radius, r_1, by taking the average of the distance between the first chord and o_1 and the distance between the second chord and o_1. The bisecting lines of the second and third chords intersect at o_2 from which r_2 can be calculated. The difference in value between r_1 and r_2 correctly indicates the radius of the path and its variance. Such radii values are clipped between 0m and 100m.

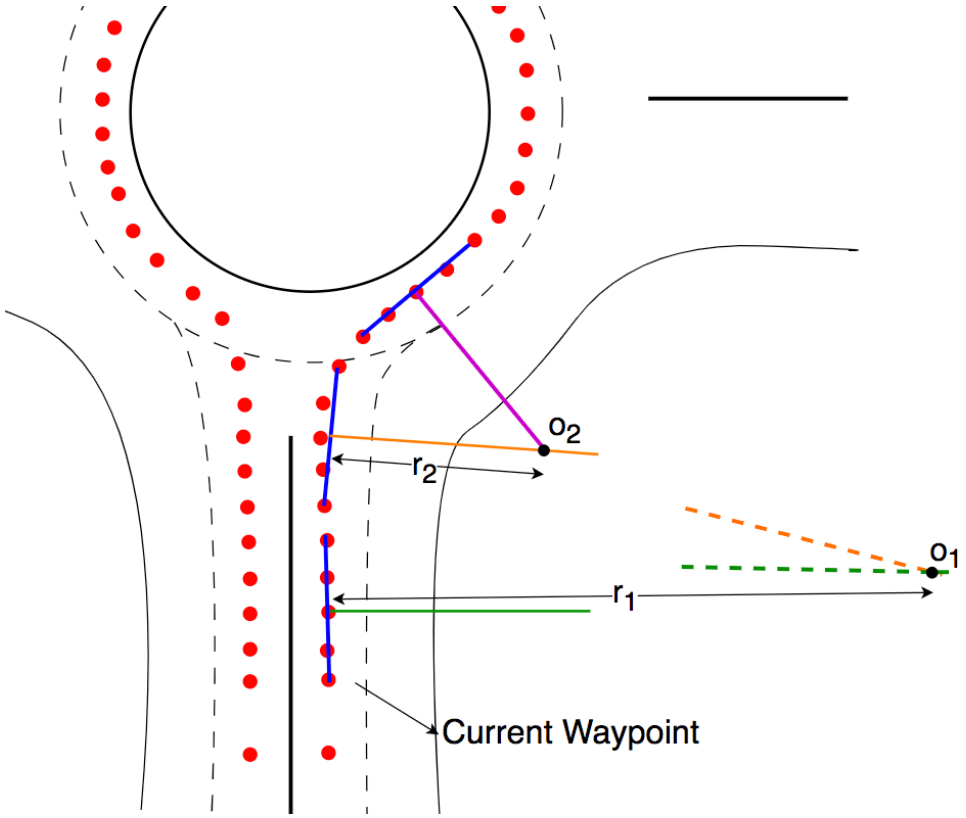


Figure 4.12 Visual illustration for calculating observation r.

While the radius offers a more general value for the curvature of the rest of the route, the angle between waypoints determined above offers a more accurate perception of the curvature for the previous and current sections. This is due to the usage of the chord while calculating the radius which adds another element of approximation. The importance of using both θ_w and r as observations was analysed by training three models which differed in the state space used. The first model only utilised the angles between waypoints, θ_w. The second model only used the different radii along the route, r, while the third model utilised both observations. Comparing such models allows us to analyse the value of using each observation.

Apart from the elements $\langle v, d_w, d_l, d_c^t \rangle$, all other distance values are normalised to a value between 0 and 1, while all angles are calculated in radians. This improves training stability and performance by having all features on a similar scale and therefore preventing the RL algorithm from giving higher importance to larger values.

Action Space

A vehicle has two primary controls: longitudinal control, acceleration/braking, and lateral control, steering. CARLA allows an acceleration value between 0 and 1, a braking value between 0 and 1, and a steering angle between -1 and 1, corresponding to a left and right steering turn. Previous research has applied both discrete [23, 30, 52] and continuous [13, 16, 17] action spaces with success. Due to the increased computational complexity incurred when employing continuous action spaces, this work will compare two discrete action spaces. The first action space, defined in Table 4.2 controls the steering angle while the acceleration value is controlled by a PI controller, maintaining a constant velocity. The second action space is defined in Section 4.4 as part of Objective 4. This action space controls the acceleration, steering, and braking values.

Table 4.2 Constant velocity action space which uses a PI controller to maintain a constant velocity of $8km/h$

Action no.	Acceleration	Steering
1	PI Acceleration Value	-0.8
2		-0.6
3		-0.4
4		-0.2
5		0.0
6		0.2
7		0.4
8		0.6
9		0.8

In this section, the first action space will be introduced. This action space maintains constant velocity and is composed of 9 discrete actions, as described in Table 4.2. The steering angle varies between -0.8 and 0.8 in steps of 0.2 while the acceleration value is controlled by a PI controller. The constants in Equation 4.2 were utilised after tuning the controller using the Ziegler-Nichols tuning method [66] in order to maintain a constant forward velocity of $8km/h$. This velocity was observed to be suitable for completing the task successfully. In practice, this action space only allowed the tractor-trailer vehicle to move forward with a chosen steering angle. No action allowed the agent to apply the brakes. The PI implementation of Lundberg[10] was utilised.

$$K_p = 0.2; K_i = 0.2 \tag{4.2}$$

As can be seen in Table 4.2 the steering angles chosen varied in steps of 0.2 while a maximum steering angle of ± 0.8 was chosen. Such restrictions provided a suitable level of granularity for controlling the vehicle while reducing the computational complexity.

[10] www.github.com/m-lundberg/simple-pid

Reward Function

Our reward function is composed of a utility function, r_u, and a reward shaping function, r_s, as defined in Equation 4.3. As mentioned by Knox et al. [60], reward shaping should be kept to a minimum and be justified since it can lead to undesirable behaviour, decreasing overall performance, despite its intention being the opposite.

$$r = r_u + r_s \tag{4.3}$$

The utility function r_u is made up of two components as defined in Equation 4.4.

$$r_u = r_w + r_e \tag{4.4}$$

r_w is a reward of $+0.1$ each time the agent passes a waypoint. r_e is the reward given to the agent at the end of each episode and is dependent on the reason why the episode was terminated as defined in Equation 4.5.

$$r_e = \begin{cases} +1 & \text{if the vehicle has arrived at the final waypoint} \\ -1 & \text{if:} \\ & \text{- there is a collision with the vehicle} \\ & \text{- the episode has exceeded 2000 timesteps} \\ & \text{- the vehicle has moved away more than 10 meters from the desired path} \end{cases} \tag{4.5}$$

The terminal state and negative reward given when the tractor-trailer vehicle is more than 10 meters away from the next waypoint was implemented to encourage the agent to move towards the end goal. The terminal state and negative reward due to the episode exceeding 2000 timesteps was implemented to prevent the agent from taking too long to complete the task.

The negative rewards of r_e help training convergence by limiting the amount of timesteps from which no successful outcome can occur. These rewards do not encourage the agent to find the shortest path between the starting and ending waypoint.

The reward shaping function, r_s, is a negative reward based on the distance between the centre of the tractor unit and the lane and is formally defined in Equation 4.6,

$$r_s = -\frac{clip(d_c^t, 0, 4)}{400} \tag{4.6}$$

where d_c^t is the distance of the tractor unit to the centre of the lane. The clipping function ensures that the distance is a value between 0 and 4. A value of 4 was chosen as the maximum since the maximum distance the tractor can be away from the centre

of the lane is 4.35 meters and clipping such value allows a higher difference in reward after normalising.

To find the optimal weight vector for the reward of the distance of the tractor unit to the centre of the lane, a number of different ratios were tested. Observing the behaviour of the vehicle, an average of 10 timesteps were performed between each waypoint. This entails that the reward shaping function, r_s, is awarded to the agent 10 times for each waypoint reward, r_w. With the average distance of the tractor to the centre of the lane being 0.6, the agent would obtain a reward of -6 for each waypoint reward of $+0.1$. The agent should be significantly more positively rewarded for moving forward and passing a waypoint rather than staying close to the centre of the lane since the overall goal of the agent is to complete the route. In order to achieve this, a weight vector is applied to r_s to reduce its value. Our initial thoughts were to reduce the negative reward to half of the positive reward obtained by passing each waypoint, by using a weight vector of $\frac{3.33}{400}$. The remaining half of the positive reward was thought to be suitable to encourage the agent to move forward while still allowing variance in the reward to encourage the agent to stay close to the centre of the lane. After running several experiments, it was identified that a significantly lower weight vector was required. Testing using weight vectors $\frac{0.5}{400}$, $\frac{1.0}{400}$, $\frac{1.5}{400}$, and $\frac{2.0}{400}$, the $\frac{1.0}{400}$ weight vector was found optimal.

Such a reward was used in several previous research [13, 26, 59] and is of high importance in our work since the tractor-trailer vehicle may opt to drive in between lanes to avoid collision on all occasions. This reward encourages the agent to drive closer to the centre of the lane when possible.

State-Transition Model

The state-transition model represents the effects of actions on a particular state [32]. In our work, this is computed by the CARLA simulator where, depending on the current forces acting on the tractor-trailer vehicle, the chosen action will produce the appropriate effect and move the vehicle to its new location. Using a high-fidelity simulator allows the state-transition model to be as realistic as possible, giving more confidence that the results achieved can be replicated in real-life scenarios.

Discount Factor

The discount factor, γ, determines the importance of future rewards when evaluating immediate actions. In this work, γ is set to 1 so that future rewards are given equal importance to current rewards. In practice, this encourages the agent to move away from the centre of the lane to avoid a collision, receiving a higher negative reward r_s, in order to achieve a greater total reward by completing the episode.

4.3 Objective 3: Comparing state-of-the-art RL algorithms

This objective aims to compare the differences in behaviour between different state-of-the-art model-free RL algorithms, in order to find the optimal RL algorithm for our autonomous driving problem.

There are different ways to differentiate between RL algorithms, but in this work, the different algorithms are chosen based on the way the agent is trained. PPO learns the optimal parameters for the policy by updating the policy only using data obtained in the latest iteration. This data is generated by acting according to the latest policy, meaning that the algorithm uses the on-policy technique. On the other hand, DQN trains the agent by learning the optimal action-value function. This training is done in an off-policy manner where the data used to optimise the action-value function is not obtained by following the current action-value function.

Comparing PPO and DQN allows us to evaluate the difference in using an on-policy, policy optimisation technique and an off-policy, q-learning technique and how this affects the behaviour of the agent. Further to this, this work will also analyse and compare the behaviour of the SAC algorithm. Despite also being off-policy, the SAC algorithm is based on the actor-critic architecture, which utilises both policy optimisation and q-learning techniques, aiming to take advantage of the benefits of both approaches. The critic updates the action-value function while the actor updates the policy parameters in the direction advised by the critic. Including such an algorithm in our comparison may indicate that a hybrid algorithm which utilises both update methods may be superior.

As mentioned in Chapter 2, utilising two q-functions in q-learning based algorithms, reduces the policy bias in the training improvement step. This improves convergence rates while also decreasing training time and therefore both the DQN and SAC algorithms will be utilising two q-functions. Furthermore, the dueling version of the double DQN model further reduces the overestimation of action values. Therefore, the dueling double DQN algorithm, DDDQN, will be used in this work.

Additionally, PPO [16], DDDQN and its variants [25, 26], and SAC [13, 27], have been successfully implemented in previous research related to autonomous navigation. All of the components mentioned in Section 4.2.2 were kept identical when testing all three algorithms. This enables a level playing field, allowing for accurate comparison. Despite this, different algorithm hyperparameters were used to ensure that each model obtained suitable convergence.

Tables 4.3, 4.4, and 4.5 define the hyperparameters used for the PPO, DDDQN, and SAC algorithm respectively. The rest of the parameters were left as default. Due to the computational complexity and therefore time complexity of training such algorithms, a formal grid search to find the optimal hyperparameter values was not feasible.

Table 4.3 Hyperparameters for PPO algorithm

Hyperparameters	Values
γ	1
learning rate	5×10^{-6}
batch size	4096
SGD minibatch size	128
Num SGD iterations	30
exploration policy	Stochastic Sampling

Table 4.4 Hyperparameters for DDDQN algorithm

Hyperparameters	Values
γ	1
double q	True
dueling	True
learning rate	5×10^{-7}
batch size	256
hidden layers	[512,512,1024]
prioritized replay buffer capacity	6×10^{5}
exploration policy	Epsilon Greedy
initial epsilon	1.0
final epsilon	0.01
exploration epsilon timesteps	1×10^{6}

Table 4.5 Hyperparameters for SAC algorithm

Hyperparameters	Values
γ	1
double q	True
actor learning rate	5×10^{-5}
critic learning rate	5×10^{-5}
entropy learning rate	5×10^{-5}
batch size	256
q model hidden layers	[512,512,1024]
policy model hidden layers	[512,512,1024]
prioritized replay buffer capacity	6×10^{5}
exploration policy	Epsilon Greedy
initial epsilon	1.0
final epsilon	0.01
exploration epsilon timesteps	1×10^{6}

Instead, the hyperparameter values were chosen from an informal search, which started with the default algorithm parameters. Despite this, several hyperparameter values were optimised during this informal search, including the learning rate, batch size, and hidden

layers. As previously mentioned, a discount factor, γ, of 1 was used in all the algorithms for the agent to give equal importance to future rewards. Furthermore, both q-learning based algorithms made use of two q-functions in favour of faster convergence.

For the PPO algorithm, the default learning rate of 1×10^{-3} was not able to obtain a converging model and was therefore lowered to 5×10^{-6}. The RLlib framework offered a similar default learning rate for the SAC model, but this was optimised to a value of 5×10^{-5} for suitable convergence. Despite providing a lower default learning rate of 5×10^{-4}, the DDDQN algorithm's learning rate was still further reduced.

A larger batch size allows for the algorithm to obtain better generalisability at the cost of the model's performance. The default batch size for the DDDQN and SAC models is 32. In the context of the number of timesteps the model was trained for, such value was deemed low and therefore the batch size was increased to 256 for both models. The batch size of the PPO model was rounded up to $2^{12} = 4096$ from a value of 4000. Experiments with larger batch sizes for the DDDQN and SAC models were tested, but this proved to significantly increase training time.

Larger hidden layers in the internal neural networks of each algorithm allow the algorithm to learn a more complex relationship between the input states and output actions. The default hidden layers for the DDDQN and SAC models are [256] and [256, 256] respectively. The complexity of these neural networks was increased to [512, 512, 1024] for both algorithms.

Since the DDDQN and SAC algorithms learn using an off-policy approach, these algorithms store the state, action, and reward experienced in previous timesteps. 6×10^5 of these previous timesteps are stored and made available to the agent, in a prioritized list, to enhance its learning from previously visited states.

The default stochastic sampling exploration policy for the PPO algorithm was utilised. The DDDQN and SAC models algorithms the Epsilon greedy exploration policy as default and therefore this was utilised with a higher Epsilon timestep of 1×10^6. At the start of training, the initial Epsilon of 1 ensures that random actions are chosen. The Epsilon value is reduced to 0.01 at 1×10^6 timesteps where random actions are only chosen 1% of the time.

4.4 Objective 4: Incorporating acceleration actions

This objective aims to compare and evaluate the constant velocity action space defined in Section 4.2.2 to an action space with allows the RL algorithm to vary its velocity through different acceleration values. This variable velocity action space allows us to evaluate the behaviour of the agent when it is able to control both the lateral and longitudinal movements of the vehicle. Such greater vehicle control may enable the agent to find a more optimal approach to solving the problem at hand.

Table 4.6 specifies each of the 29 discrete actions which compose the variable velocity action space. The first action allows the agent to perform no steering, acceleration, or braking control on the vehicle. The second action allows the agent to apply the brake. The remaining 27 actions allow the agent to apply an acceleration value of $0.1, 0.3$, or 0.6, along with a steering angle of $-0.8, -0.6, -0.4, -0.2, 0.0, 0.2, 0.4, 0.6,$ or 0.8. Such an action space enables more accurate control of the vehicle when compared to the constant velocity action space.

Table 4.6 Variable velocity action space using 4 different acceleration values

Action no.	Acceleration	Steering	Brake
1	0.0	0.0	0.0
2	0.0	0.0	1.0
3-11	0.1	[-0.8, -0.6, -0.4, -0.2, 0.0, 0.2, 0.4, 0.6, 0.8]	0.0
12-20	0.3		0.0
21-29	0.6		0.0

When using the variable velocity action space, the agent may continuously choose actions 1 and/or 2, as defined in Table 4.6. Action 1 applies no control to the vehicle, allowing for coasting, while action 2 applies the brake. If these actions are continuously chosen, the tractor-trailer vehicle would not be moving towards the next waypoint. Therefore, if the vehicle stays stationary for more than 100 timesteps, the episode is terminated and a negative reward of -1 is given to the agent. Reward r_e is now updated to be defined as Equation 4.7 and is used when using the variable velocity action space.

$$r_e = \begin{cases} +1 & \text{if the vehicle has arrived at the final waypoint} \\ -1 & \text{if:} \\ & \text{- there is a collision with the vehicle} \\ & \text{- the episode has exceeded 2000 timesteps} \\ & \text{- the vehicle has moved away more than 10 meters from the desired path} \\ & \text{- the vehicle has remained stationary for more than 100 timesteps} \end{cases} \tag{4.7}$$

While performing experiments using the variable velocity action space, a large variation in velocity throughout the episode was observed. To mitigate this undesirable behaviour and encourage a more stable velocity, a negative reward based on the change in velocity is given to the agent. This is considered as part of the reward shaping function, r_s and therefore when using the variable velocity action space, the reward shaping function is defined by:

$$r_s = -\frac{clip(d_c^t, 0, 4)}{400} - \frac{5 \times clip(|\Delta_v|, 0, 1.2)}{120} \tag{4.8}$$

where, Δ_v is the difference in velocity from the previous to the current timestep. Observing the difference in velocities collected during multiple episodes, a maximum difference in velocity of 1.2 was determined to be a suitable value for using as a maximum of the clip function. This captures the exact value of the differences in velocities in most cases while capping larger differences at 1.2. Limiting the possible values using the clip function allows for a higher difference in the reward between small values of velocity. The clipped value is then normalised to a value between 0 and 0.01, by dividing by 120. The resulting value is multiplied by 5 since, after some experimentation, a weight vector of $\frac{5}{120}$ was deemed suitable when considering other rewards and that this reward is given at each timestep. This weight vector provides a suitable negative reward to the agent to encourage more stable velocity, while not discouraging the agent from reaching higher velocity values.

4.5 Objectives 2, 3, and 4: Training Setup

Training of all RL models was performed using the RLlib framework and their implementations of the PPO, twin-q SAC, and dueling double DQN algorithms. A total of 52 routes were used during training, which included all possible route variations when navigating a roundabout and were randomly chosen from the 16m, 32m and 50m roundabouts defined in Section 4.1.3. Training time varied between 8 and 24 hours, generating between 3.5 million and 20 million timesteps. The longer training time is due to the increased number of actions when training models using the variable-velocity action space. Furthermore, the DDDQN and SAC models took a considerably longer time to converge despite using 6 simulators in parallel. Such significant training times limited the amount of hyperparameter tuning that could be performed. A higher learning rate could be used to decrease training time at the cost of instability and convergence.

Training was performed on an Intel Core i7-12700K with 64GB of RAM and a GeForce GTX3060 GPU running Ubuntu 18.04. Development, training, and testing were implemented using Python version 3.8, RLlib version 2.5, and CARLA version 0.9.13. A timestep of 0.1, 10Hz, was used for the CARLA simulator based on previous research.

5 Evaluation

In this chapter, different evaluation strategies will be used to present the results obtained from the experiments described in the previous chapter. For objective 1, the developed tractor-trailer model and custom roundabout scenarios will be tested to ensure resemblance to real-life behaviour. In objective 2, different RL models using different state, action, and reward spaces will also be compared.

5.1 Objective 1: Simulation Environment

The developed tractor-trailer model is tested to ensure that it correctly resembles a real-life vehicle. This includes testing its movement when throttle, brake, and steering actions are applied. Throttle and braking actions were performed both on the tractor unit alone and on the tractor-trailer vehicle. The acceleration and deceleration values of the tractor unit are considered to resemble real-life behaviour since the engine power, weight, and braking forces were modified in CARLA to match the manufacturer's specifications[1]. Figure 5.1 illustrates how the velocity of the tractor unit alone and the velocity of the tractor-trailer vehicle varies when full throttle is applied. Due to the increased weight of the trailer unit, a slower increase in velocity is observed for the tractor-trailer vehicle. This is illustrated by the orange curve being under the blue curve, indicating that at the same timestep, the tractor unit alone had a higher velocity. The parts of the graphs which show constant velocity signify when gear changes were occurring. Furthermore, the peak velocity of 30km/h is achieved 150 timesteps earlier by the tractor unit only.

Similar behaviour can be observed when analysing the changes in velocity under deceleration, shown in Figure 5.2. The tractor-trailer vehicle is noted to decrease its velocity slower. At the same timestep, the tractor unit is observed to have a lower velocity when compared to that of the tractor-trailer vehicle. This is due to the trailer's momentum, which increases the braking effort required to stop the vehicle. Such behaviour is expected and confirms that the physical properties of the trailer unit and the implications that arise from having a trailer connected are being taken into consideration.

Steering actions were also performed on the tractor unit alone, ensuring that the turning circle between walls is 15.38m, as defined by the manufacturer specifications for a similar tractor unit, the DAF XF 450 FT[1]. The observed turning circle between walls was 15.20m and therefore such property can be said to be almost identical to the expected value. This was performed both for full left and full right turns. Such comparable values give confidence that the steering dynamics of the simulated vehicle correctly resemble real-life behaviour.

[1] `www.daf.co.uk/en-gb/trucks/specsheets-search-page`

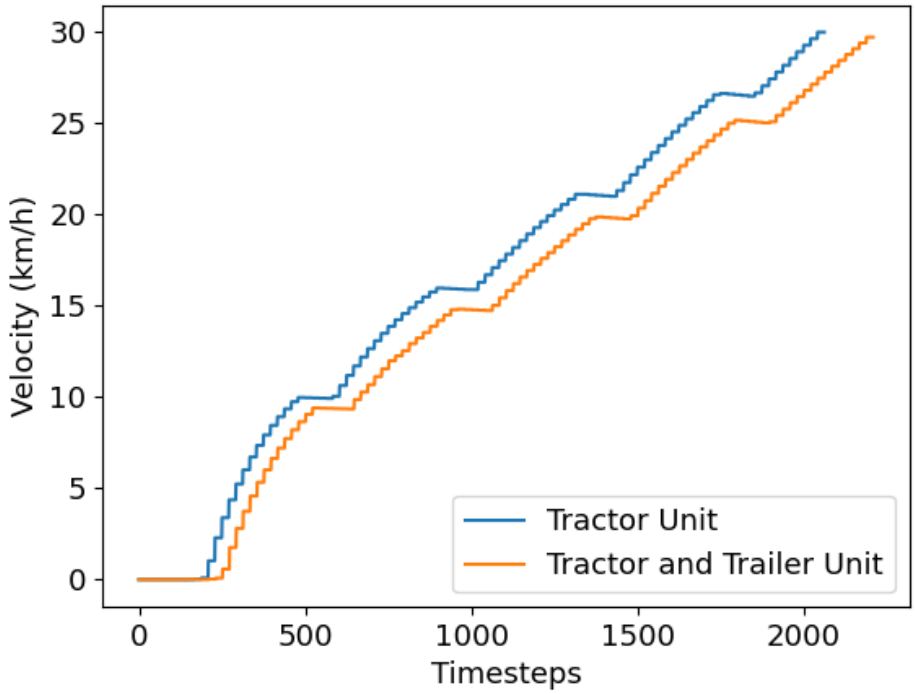


Figure 5.1 Velocity changes under acceleration for the tractor unit alone and the tractor-trailer vehicle.

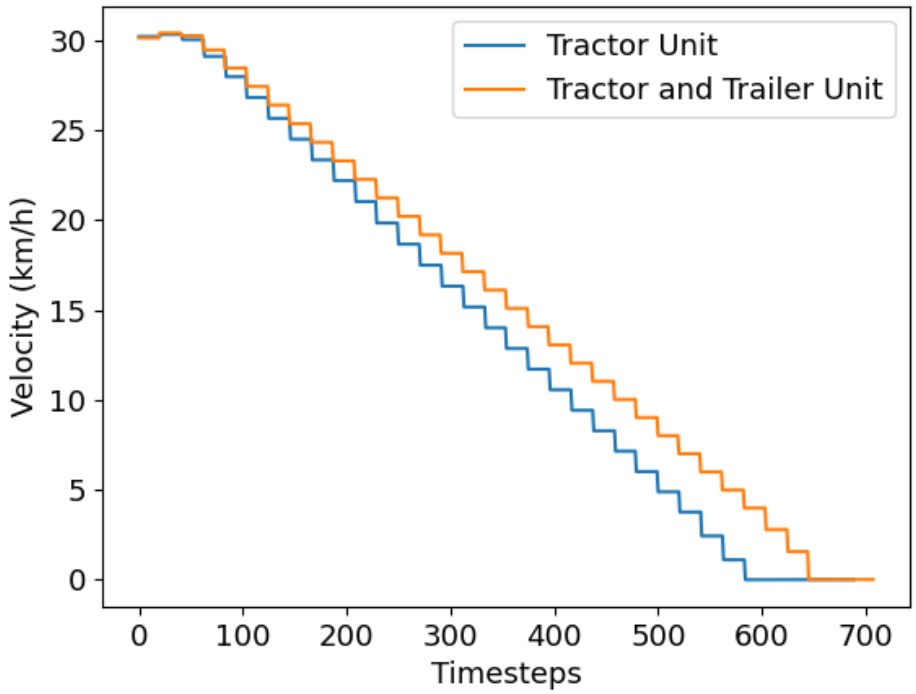


Figure 5.2 Velocity changes under deceleration for the tractor unit alone and the tractor-trailer vehicle.

From performing steering actions with the trailer unit attached, one can observe the correct relative movement between the tractor and trailer units. The articulation point joining the tractor and trailer unit allows both units to be independent from each other in terms of heading while still being connected together. This is illustrated in Figure 5.3 where the tractor and trailer units are connected to each other but are at different angles.

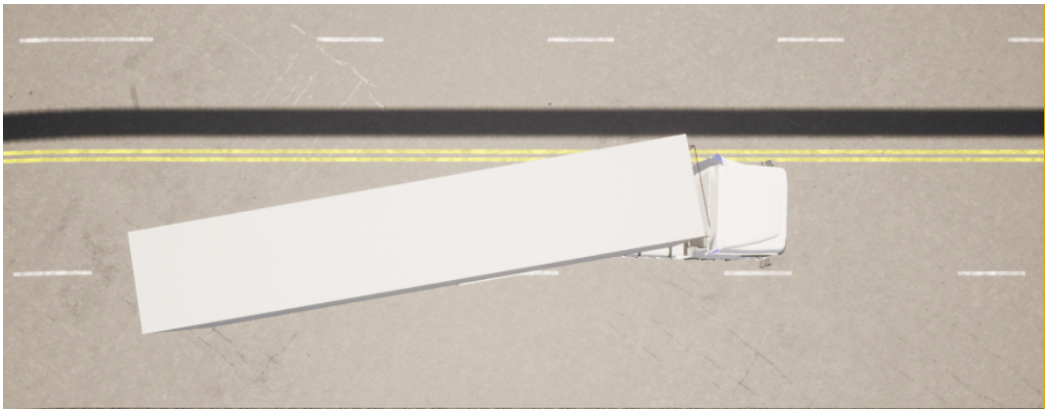


Figure 5.3 Proper functioning of the articulated joint between the tractor and trailer units.

The correct functioning of the distance sensors was confirmed by placing the vehicle at various positions throughout the environment and observing the obtained values. Figure 5.4 illustrates the values obtained by the 29 distance sensors placed throughout the tractor-trailer vehicle. One can observe that the distance value correctly increases and decreases depending on the position of the kerb. It should be noted that, as mentioned in Section 4.2.2, such distance sensors have a maximum range of 7.0m and are normalised to values between 0 and 1. Therefore, the 1.0 values marked in Figure 5.4 indicate that the road boundary is greater than 7m away from the respective sensor.

We also ensure that collisions are detected by the tractor and trailer units at various positions and angles through trial and error. Correct weight and vehicle dimensions for both the tractor and trailer units were also confirmed, as mentioned in Section 4.1.2.

With regards to the developed roundabout scenarios, each of the possible routes the vehicle may be trained or tested on was traversed using a manually controlled tractor-trailer vehicle. This ensures that all routes are adequately designed such that the tractor-trailer vehicle can complete the route without any collisions. The accurate detection of collisions with the kerb was also tested at different positions and angles of each roundabout.

After testing the tractor-trailer vehicle and the roundabout scenarios, no clear inaccuracy was detected, and both models were observed to adequately resemble their real-life counterparts.

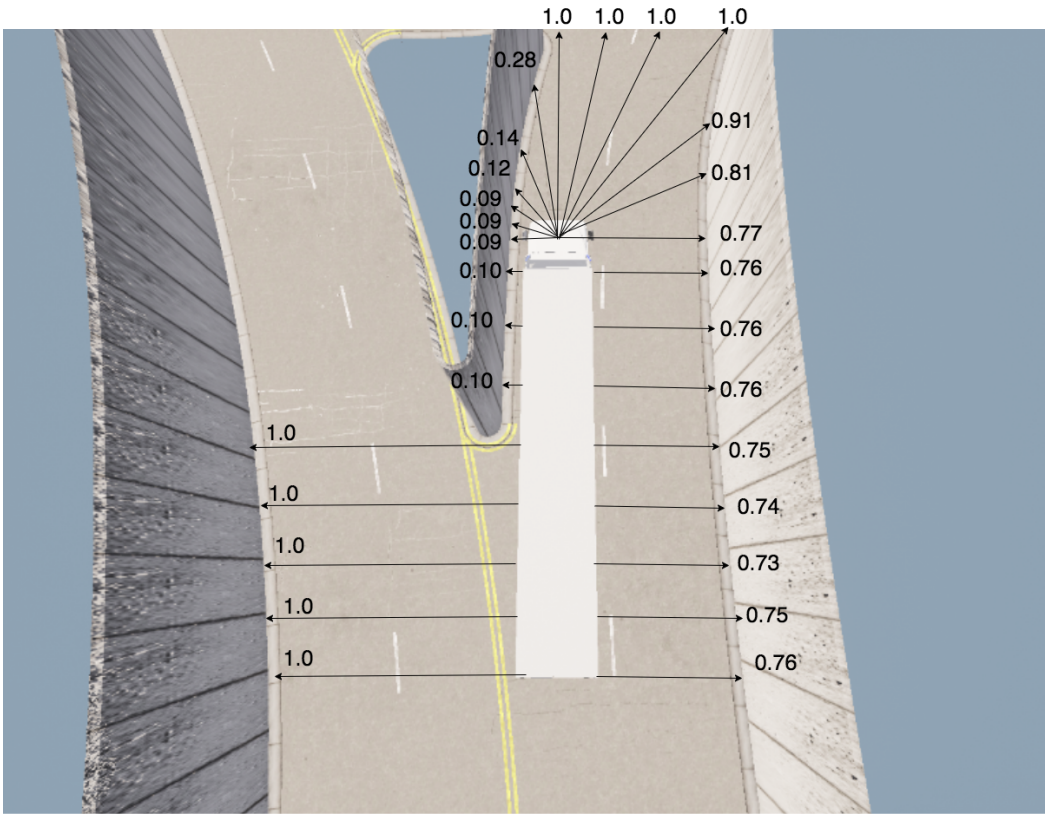


Figure 5.4 Distance values computed by the 29 distance sensors.

5.2 Objective 2: Reinforcement Learning Environment

Testing the RL models was performed on the 20m and 40m diameter roundabouts. These were not used during training and have different physical properties than the roundabouts used during training. This allows for better evaluation and can more accurately indicate the generalisability of the model. The testing routes include both larger and smaller radii turns. A total of 23 different routes were tested, each run 5 times, resulting in 115 testing episodes for each model. Given that training was performed on 52 routes, a train-test ratio of 0.7/0.3 is utilised.

7 metrics were utilised to quantitatively analyse the generated models. These are success rate, tractor collision rate, trailer collision rate, truck distance to the centre of the lane, d_c^t, trailer distance to the centre of the lane, d_c^{tt}, timesteps, and timeout rate.

The success rate is defined by Equation 5.1. e_{total} is the total number of episodes.

$$\text{success rate} = \frac{e_{success}}{e_{total}} \tag{5.1}$$

where $e_{success}$ is the number of episodes in which the vehicle reached the final waypoint.

The tractor collision rate is defined by Equation 5.2.

$$\text{tractor collision rate} = \frac{e_{tractor}}{e_{total}} \tag{5.2}$$

where $e_{tractor}$ is the number of episodes in which the tractor unit collided with the environment.

The trailer collision rate is defined by Equation 5.3.

$$\text{trailer collision rate} = \frac{e_{trailer}}{e_{total}} \tag{5.3}$$

where $e_{trailer}$ is the number of episodes in which the trailer unit collided with the environment.

Both the distance between the centre of the tractor unit and the trailer unit to the centre of the lane will be noted. Both metrics are essential to evaluate the models, where the value of such metrics should be minimised. Having both metrics allows us to ensure that the agent has navigated the vehicle as close to the centre of the lane as possible and only when required to avoid collision, has the tractor unit moved away from the centre of the lane. This is true when the distance between the trailer and the centre of the lane is minimised. On the other hand, when both distances to the centre of the lane are relatively large, the tractor unit would have navigated away from the centre of the lane well before required, essentially increasing the curvature of the path and therefore decreasing the complexity of the turn, at the cost of increasing both distances to the centre of the lane which may result in the vehicle crossing into other lanes which is undesirable behaviour. Timesteps are the average total number of ticks the simulation performs for each successful episode while the timeout rate is defined by Equation 5.4.

$$\text{timeout rate} = \frac{e_{timeout}}{e_{total}} \tag{5.4}$$

where $e_{timeout}$ is the number of episodes which exceed 2000 timesteps.

As discussed in Chapter 3, another metric to evaluate the navigation of a tractor-trailer vehicle is the centring of the swept area inside the road boundary [15]. Having a singular metric rather than two metrics, the distances of both the tractor and trailer units to the centre of the lane would be advantageous, but unfortunately, such a metric cannot be applied to our work. Ensuring that the swept area by the tractor-trailer vehicle is equidistant to the road boundary requires a hard boundary which cannot be passed. Such hard boundaries can be obtained in one-lane roundabouts where the width and curvature of the lane allow the tractor-trailer vehicle to be navigated through, as in the testing scenarios used by Oliveira et al. [15]. In our two-lane roundabout scenarios, the lower-diameter roundabouts force the tractor-trailer vehicle to cross into other lanes in order to complete the turn without any collisions. Therefore, the only possible hard boundaries would be the inner and outer kerb of the roundabout. If these were used,

a model would be penalised if the agent navigated the vehicle as close to the centre of the appropriate lane as possible whereas a model where the tractor-trailer vehicle is in between the two lanes, which is undesirable behaviour, would be favoured. Therefore, in our case, such a metric cannot be utilised to correctly evaluate the developed models.

At the time of writing, no research focuses on the lateral and longitudinal control of tractor-trailer vehicles through roundabout intersections using an RL framework. Despite several research successfully generating models for light passenger vehicles with such control, the different physical characteristics and constraints of tractor-trailer vehicles prevent the use of such models and their underlying structure. In order to obtain a suitable RL environment, specifically an appropriate state space and reward function, along with satisfactory RL algorithm hyperparameters, countless variants were tested.

A number of variants were evaluated and compared as follows:

1. Different weight vectors for the distance of the tractor unit to the centre of the lane in the reward function.

2. The effect of using the radius and/or angle between waypoints as part of the observation space.

A trajectory analysis of the constant velocity action space model was also performed to visually analyse the model's behaviour. Finally, the results obtained will be compared to those observed in current literature.

5.2.1 Different weight vectors for the distance of the tractor unit to the centre of the lane in the reward function

As discussed in Section 4.2.2, 4 different weight vectors for the reward of the distance of the tractor unit to the centre of the lane, r_s, were tested. Such weight vectors are $\frac{0.5}{400}$, $\frac{1.0}{400}$, $\frac{1.5}{400}$, and $\frac{2.0}{400}$. It should be noted that all the models compared utilised the same hyperparameters, while only the reward function was modified. Furthermore, the constant velocity action space was utilised, which also entails that no episodes timed out since each action in this action space has a positive acceleration.

Figures 5.5 and 5.6 illustrate the average total discounted reward per episode obtained by each model during training. These models differ in the weight vector used for the reward of the distance between the tractor unit and the centre of the lane. The label in each graph denotes the weight vector of the reward shaping function r_s. As can be observed in Figure 5.5 the model with weight vector $\frac{1.0}{400}$, green graph, obtains the highest average return. This is more clearly observed when using a sliding window of 1500, rather than 200, as shown in Figure 5.6. Going from highest to lowest average return, the remaining models fair as follows $\frac{0.5}{400}$, $\frac{1.5}{400}$, and $\frac{2.0}{400}$.

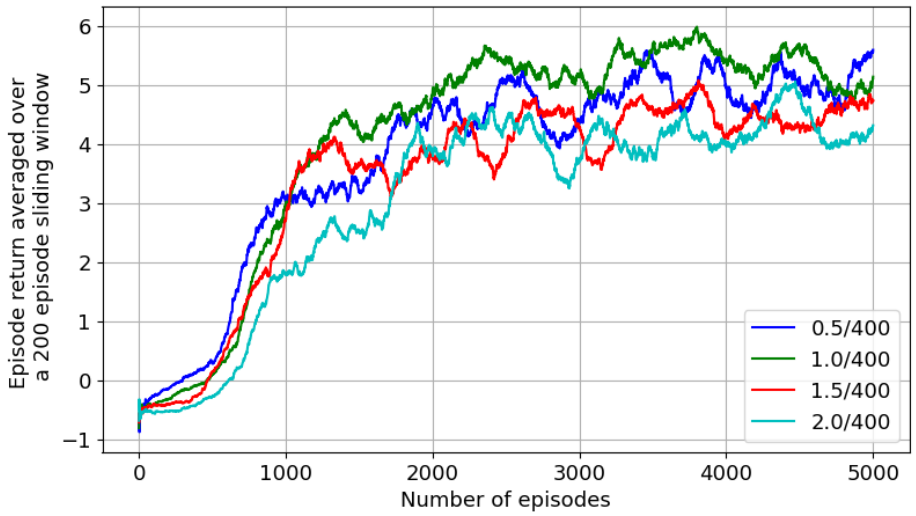


Figure 5.5 Average training return graph for models using different weight vectors for the distance of the tractor unit to the centre of the lane averaged over a 200 episode sliding window.

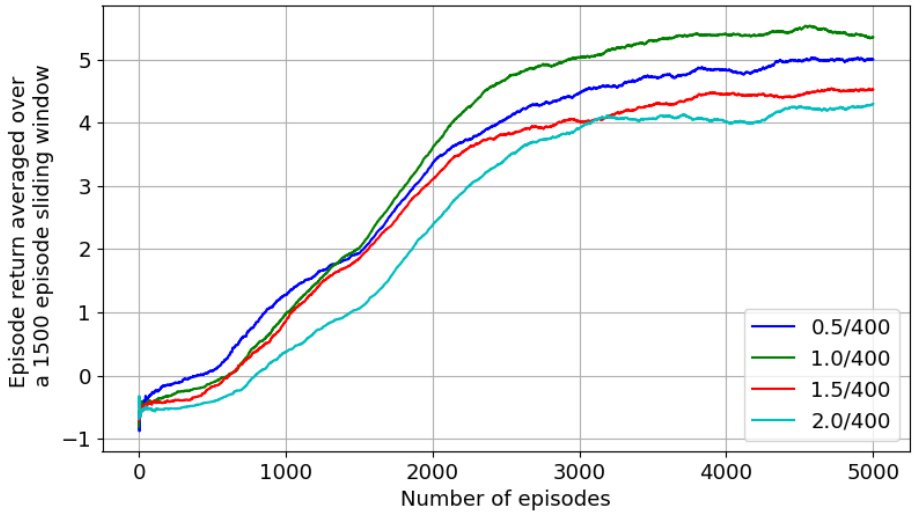
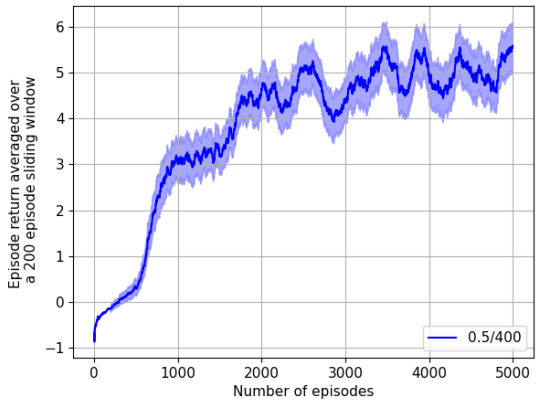
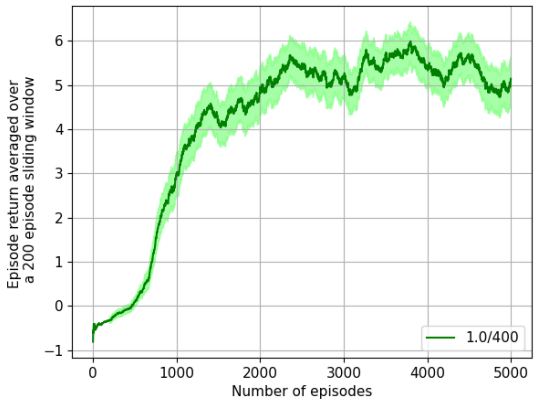


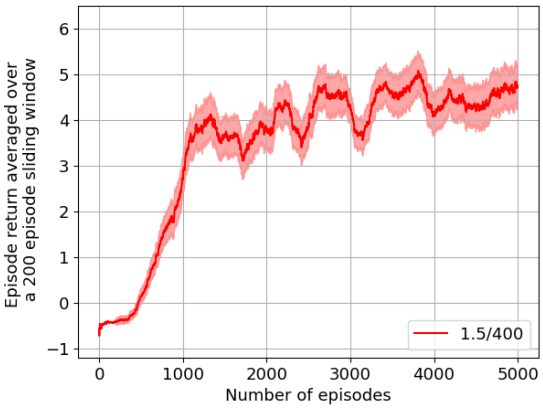
Figure 5.6 Average training return graph for models using different weight vectors for the distance of the tractor unit to the centre of the lane averaged over a 1500 episode sliding window.



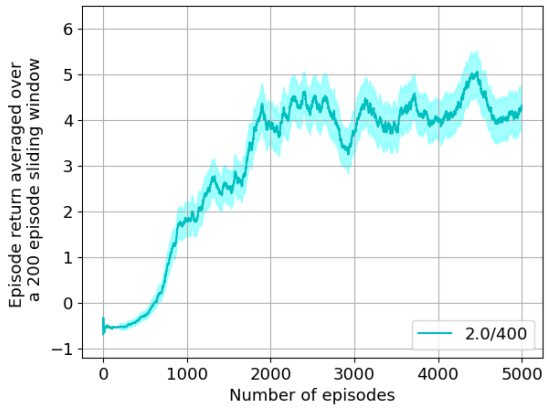
(a) $\frac{0.5}{400}$ model



(b) $\frac{1.0}{400}$ model



(c) $\frac{1.5}{400}$ model



(d) $\frac{2.0}{400}$ model

Figure 5.7 Average training return graphs for models using different weight vectors for the distance of the tractor unit to the centre of the lane averaged over a 200 episode sliding window. The shaded area represents the 95% confidence interval.

Figure 5.7 illustrates the average return obtained by each model, averaged over a 200 episode sliding window, where the shaded area represents the 95% confidence interval. A significant amount of variance in the return can be observed. This could indicate that the policy has yet to reach its peak convergence.

A 200-episode sliding window was chosen to average the training return since such a value was found to smooth out the data while still allowing for changes in the return to be observable. A 1500-episode sliding window was chosen since such a value obtains a more general outlook of the model's return value.

Due to the ± 1 reward, r_e, given to the agent based on whether the episode has been successfully completed or not, the average return for the model is expected to increase if more episodes are successfully completed. The average return is also negatively influenced by the tractor's distance to the centre of the lane. This entails that if all the models in Figure 5.6 obtained the same success rate and maintained a similar distance away from the centre of the lane, the average return would be slightly higher from one model to another when going from a higher to a lower weight vector for r_s, since this is a negative reward. Based on this, if all models obtained the same success rate and maintained a similar distance to the centre of the lane, the following order going from highest to lowest average return would be observed: $\frac{0.5}{400}$, $\frac{1.0}{400}$, $\frac{1.5}{400}$, $\frac{2.0}{400}$. Since the model using the weight vector $\frac{1.0}{400}$ obtains an average return higher than that of the $\frac{0.5}{400}$, this entails that a higher success rate was obtained by the $\frac{1.0}{400}$ model, assuming that the difference between the distances to the centre of the lane was marginal. This is confirmed when inferring the obtained models on the training and testing roundabout scenarios, where the greater return obtained from a significantly higher success rate outweighs the slightly higher negative reward obtained by the tractor being further away from the centre of the lane. These metrics are shown in Tables 5.1 and 5.2.

Table 5.1 Evaluation metrics on training roundabout scenarios for models using different weight vectors for the distance of the tractor unit to the centre of the lane

Training Roundabout Scenarios					
Model	**Success Rate**	**Tractor Collision Rate**	**Trailer Collision Rate**	d_c^t	d_c^{tt}
0.5 / 400	0.62	0.22	0.17	0.73	0.60
1.0 / 400	0.81	0.05	0.14	0.61	0.43
1.5 / 400	0.62	0.09	0.29	0.42	0.42
2.0 / 400	0.66	0.1	0.23	0.42	0.32

While the $\frac{0.5}{400}$, $\frac{1.5}{400}$, and $\frac{2.0}{400}$ weight vector models obtain a training success rate between 0.62 and 0.66, the $\frac{1.0}{400}$ weight vector model obtains a significantly higher success rate of 0.81. The reason for this higher success rate can be attributed to the agent learning to deviate the tractor unit away from the centre of the lane in order for the trailer to avoid collision with the inner kerb, for tighter radius paths. This is observed

by the relatively high tractor distance to the centre of the lane, d_c^t in Table 5.1, for the model $\frac{1.0}{400}$, when compared to that of the models $\frac{1.5}{400}$ and $\frac{2.0}{400}$, where the trailer distance, d_c^{tt}, is quite similar between these three models.

Visually observing the generated model, the $\frac{1.5}{400}$ and $\frac{2.0}{400}$ models, successfully complete the routes which do not require the tractor-trailer vehicle to deviate from the centre of the lane due to their large curvature. Despite this, these models fail to navigate through lower curvature routes where the tractor-trailer vehicle fails to deviate from the centre of the lane and therefore the trailer collides with the inner kerb. This can also be observed by the high trailer collision rates of the $\frac{1.5}{400}$ and $\frac{2.0}{400}$, 0.29 and 0.23, when compared to that of the $\frac{1.0}{400}$, 0.14, model in Table 5.1. Therefore, we can conclude that the $\frac{1.5}{400}$ and $\frac{2.0}{400}$ models are not sufficient for solving the general problem since they reward the agent too negatively when deviating away from the centre of the lane.

Observing the tractor and trailer unit distances to the centre of the lane, d_c^t and d_c^{tt}, metrics in Table 5.1 for the $\frac{0.5}{400}$ model, one can note that these are higher than those obtained by the $\frac{1.0}{400}$ model. Despite these large values, a lower success rate is still achieved. Furthermore, observing the tractor and trailer collision rates for all models in Table 5.1, the tractor collision rate for the $\frac{0.5}{400}$ model is significantly higher than that of the other models, where the trailer collides in most cases.

Visually analysing the $\frac{0.5}{400}$ model while navigating through different routes, reveals that when trying to navigate through low curvature paths, the tractor unit stays in the centre of the lane much longer than the other models, after which the tractor steers completely in the appropriate direction. The roundabout is only made up of two, 3.7m wide lanes, which does not provide enough space for the tractor-trailer vehicle to complete the turn, resulting in a head-on collision with the tractor unit. In contrast, the agent in the other models deviates earlier from the centre of the lane, allowing for smaller steering angles to be used to complete the turn while still keeping the trailer unit as close to the centre of the lane as possible.

Table 5.2 Evaluation metrics on testing roundabout scenarios for models using different weight vectors for the distance of the tractor unit to the centre of the lane

Testing Roundabout Scenarios					
Model	**Success Rate**	**Tractor Collision Rate**	**Trailer Collision Rate**	d_c^t	d_c^{tt}
0.5 / 400	0.44	0.54	0.04	0.62	0.50
1.0 / 400	0.77	0.02	0.22	0.58	0.40
1.5 / 400	0.38	0.05	0.57	0.44	0.46
2.0 / 400	0.65	0.03	0.31	0.48	0.34

Observing the evaluation metrics on the testing roundabout scenarios in Table 5.2 the same conclusion can be deduced. Obtaining a 0.81 success rate, the $\frac{1.0}{400}$ model outperforms all of the other models, while a difference in the success rate of only 0.04,

when compared to its training success rate, indicates that that the model did not overfit. Similar to the training metrics, the unsuccessfully completed episodes of the $\frac{0.5}{400}$ model, are mostly due to the tractor unit colliding with the environment. Comparing the training and testing success rates of the $\frac{0.5}{400}$ and $\frac{1.5}{400}$, one can notice a significant difference which would indicate that the models are overfitting to the training roundabout scenarios.

As stated in Section 4.2.2, our initial thoughts were to provide an average negative reward of -0.05 for each positive waypoint reward of $+0.1$, by using a weight vector of $\frac{3.33}{400}$. In practice, a smaller weight vector of $\frac{1}{400}$ provided superior results. This awards the agent an average negative reward of -0.015 for each positive reward of $+0.1$. This ensured that the agent was encouraged to move forward to complete the task at hand for low and high curvature paths, while also staying as close to the centre of the lane as possible. It should be noted that such weight vector provides the optimal ratio between the negative distance reward and positive waypoint reward and can therefore be utilised irrespective of the RL algorithm.

5.2.2 The effect of using the radius and/or angle between waypoints as part of the observation space

As described in Section 4.2.2, to represent the curvature of the route, both the radius and the angle between waypoints are provided as a state input. At first glance, including both of these observations may seem redundant due to the similar information that they provide. To analyse the importance of each observation, three models were trained which differed in the state space used. The first model utilised the angles between waypoints. The second model used the different radii along the route, while the third model utilised both observations. These models are named 'Angles', 'Radii', and 'Angles and Radii' respectively. Algorithm hyperparameters and reward functions were identical for each model. The constant velocity action space and the $\frac{1.0}{400}$ weight vector were utilised.

Figures 5.8 and 5.9 illustrate the mean training return of the three models using different state spaces, averaging at two different sliding windows. In Figure 5.8, and more clearly defined in Figure 5.9, one can observe that the 'Angles' model obtains the lowest mean return. The 'Radii' and 'Angles and Radii' models are observed to reach similar average returns where the 'Angles' model reaches a value of 5 while the 'Angles and Radii' model reaches a value of 5.2. Figure 5.10 illustrates the average return obtained by each model, averaged over a 200 episode sliding window, where the shaded area represents the 95% confidence interval. A significant amount of variance in the return obtained between each episode can be observed. This could indicate that the policy has yet to reach its peak convergence. Further evaluation metrics are obtained by inferencing the developed models on the training and testing roundabout scenarios, shown in Table 5.3 and 5.4.

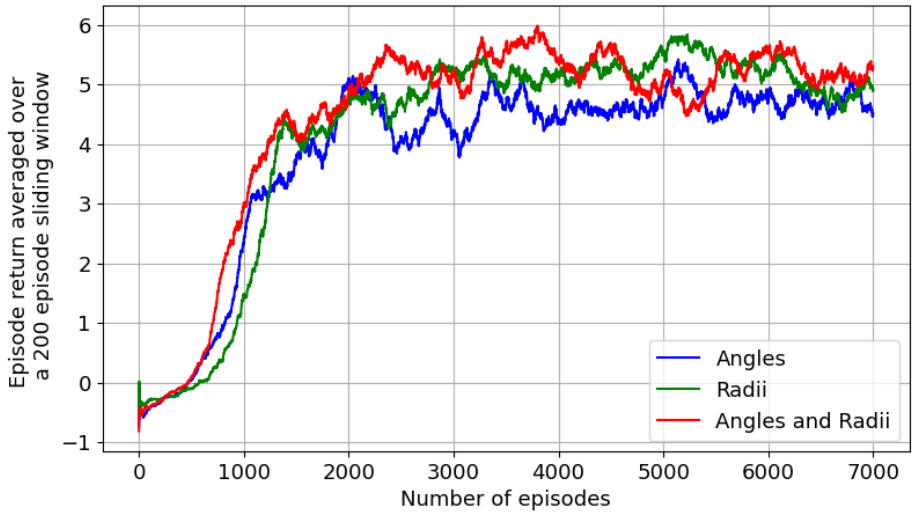


Figure 5.8 Average training return graph for the 'Angles', 'Radii', and 'Angles and Radii' models averaged over a 200 episode sliding window.

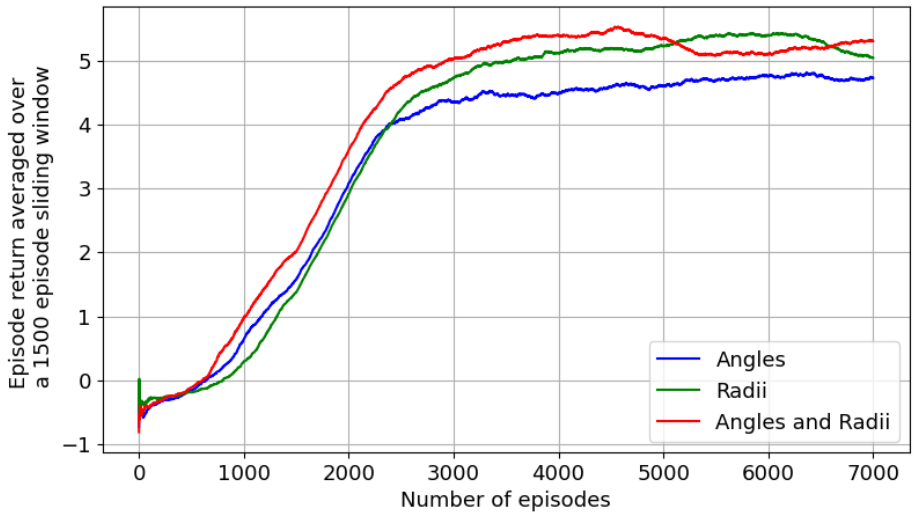
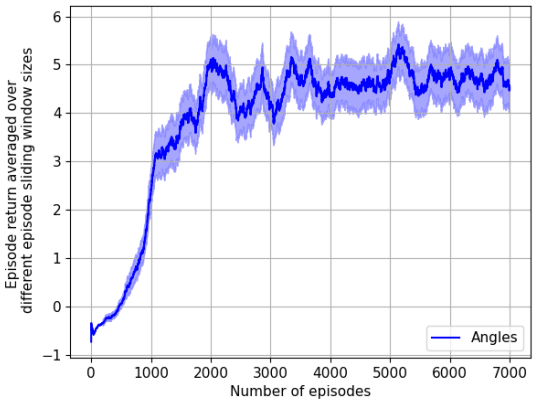
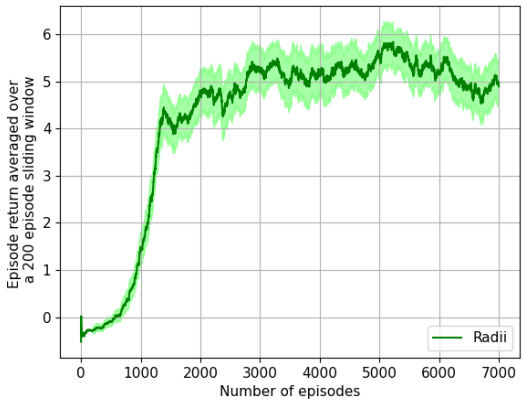


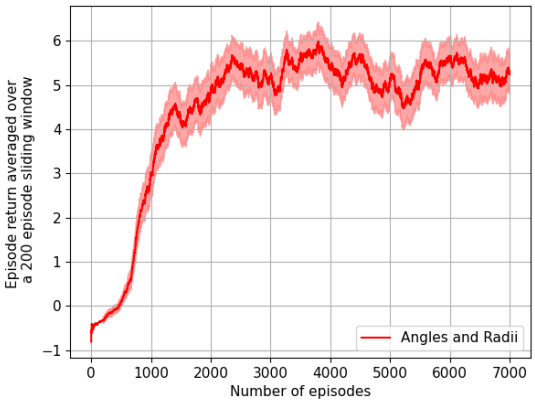
Figure 5.9 Average training return graph for the 'Angles', 'Radii', and 'Angles and Radii' models averaged over a 1500 episode sliding window.



(a) 'Angles' model



(b) 'Radii' model



(c) 'Angles and Radii' model

Figure 5.10 Average training return graphs for models using different state observations over a 200 episode sliding window. The shaded area represents the 95% confidence interval.

The success rate obtained by the 'Angles' model differs by 0.2 from the other two models. Such a lower success rate can be attributed to the fact that the agent does not sufficiently deviate away from the centre of the lane. After visually analysing the developed model, it was observed that the agent only just fails to successfully navigate the paths which require the vehicle to deviate away from the centre of the lane. Only a slightly higher deviation from the centre of the lane was required to complete the task. This is congruent with the metrics obtained in Table 5.3 where the 'Angles and Radii' model, which obtains a significantly higher success rate, only navigates the tractor unit 0.1m further away from the centre of the lane.

As discussed in Section 5.2.1 the average return is based on the end state of the agent and the distance to the centre of the lane. As observed and discussed above, the mean return for the 'Radii' and 'Angles and Radii' models are almost equivalent, with the 'Angles and Radii' model obtaining a marginally higher return. From Table 5.3 one can observe that the training success rates only differ by 0.01. Despite this, the values of d_c^t and d_c^{tt} are lower for the 'Angles and Radii' model by 0.24m. This entails that the difference in average return observed above was due to the difference in the distance of the tractor unit to the centre of the lane.

Table 5.3 Evaluation metrics on training roundabout scenarios for the 'Angles', 'Radii', and 'Angles and Radii' models

Training Roundabout Scenarios					
Model	**Success Rate**	**Tractor Collision Rate**	**Trailer Collision Rate**	d_c^t	d_c^{tt}
Angles	0.62	0.2	0.17	0.51	0.43
Radii	0.82	0.02	0.15	0.85	0.67
Angles and Radii	0.81	0.05	0.14	0.61	0.43

Evaluating the models on the testing roundabout scenarios, show in Table 5.4, the 'Angles and Radii' model obtained superior results. In addition to having a higher success rate, the tractor and trailer units maintain a closer distance to the centre of the lane by around a 0.3m difference. Furthermore, comparing the training and testing success rates, one can conclude that the 'Angles' model is overfitting to the training data due to the large difference of 0.12 obtained between the training and testing success rates. This may be due to the fact that the state observations provided in this model are not sufficient for generalising the roundabout scenario.

After evaluating the above metrics, it can be concluded the 'Angles and Radii' model is better suited to represent the problem at hand. Using the angles between waypoints and the radii of the path as part of the state space, performs superiorly in terms of success rate and distance to the centre of the lane when compared to using solely the angles to the waypoint or the radii of the path.

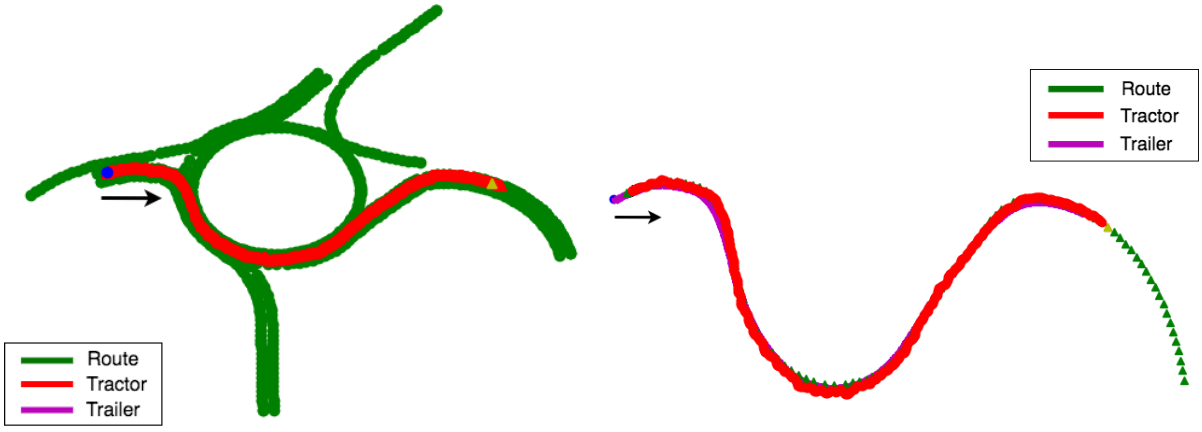
Table 5.4 Evaluation metrics on testing roundabout scenarios for the 'Angles', 'Radii', and 'Angles and Radii' models

Testing Roundabout Scenarios					
Model	**Success Rate**	**Tractor Collision Rate**	**Trailer Collision Rate**	d_c^t	d_c^{tt}
Angles	0.57	0.15	0.30	0.63	0.41
Radii	0.7	0	0.3	0.89	0.67
Angles and Radii	0.77	0.02	0.22	0.58	0.40

5.2.3 Trajectory analysis of the constant velocity action space model

To better understand the complexity of the problem at hand and how the agent navigates the tractor-trailer vehicle, an in-depth trajectory analysis of the developed constant velocity action space model was performed. In light of the previously obtained results, the evaluated model will make use of both the angles between waypoints and radii along the path, as state observations. Furthermore, the weight vector $\frac{1}{400}$ will be used for the reward shaping function r_s.

In Figures 5.11 and 5.12, the green path represents the route provided to the agent, the red path represents the path taken by the tractor unit, while the magenta path represents the path taken by the rear of the trailer unit. The rear of the trailer is defined at the centre of the 3 rear trailer axles. The blue circle represents the starting point, while the yellow triangle represents the endpoint. The black arrow represents the direction of the tractor-trailer vehicle.



(a) Route in context of the whole roundabout (b) Close up of route

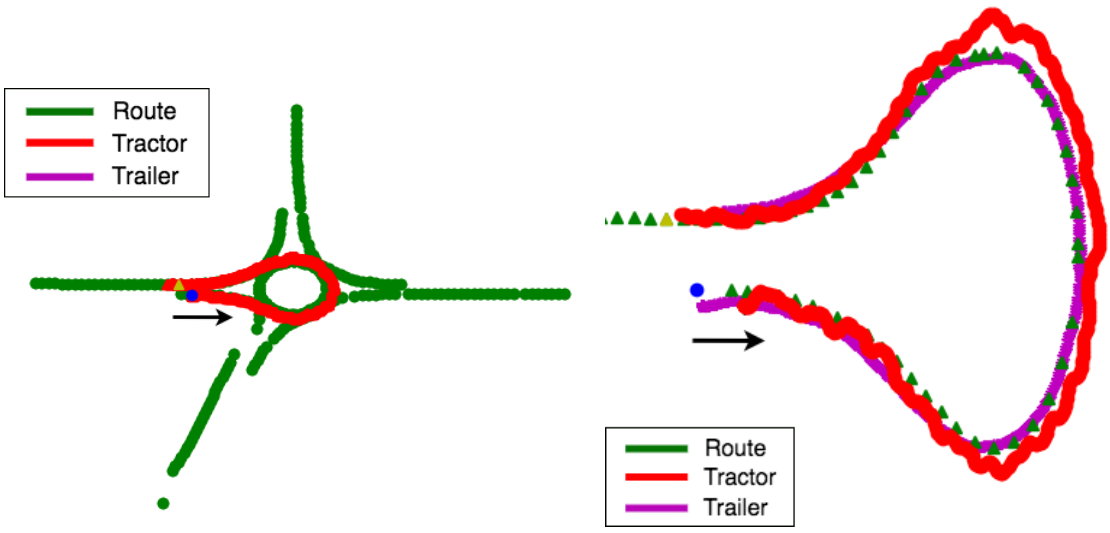
Figure 5.11 Route followed in the 50m training roundabout scenario.

Figure 5.11 illustrates one of the training routes from the 50m roundabout which does not require the tractor-trailer to deviate away from the centre of the lane in order to successfully complete the route. As can be seen in Figure 5.11b, the vehicle closely

follows the route, with only a slight deviation from the centre of the lane at the bottom of the roundabout. The trailer also closely follows the centre of the lane, since the magenta path mostly overlaps the green waypoints. Furthermore, the path the tractor unit navigates through is observed to be smooth, without any haphazard steering movements.

By observing the green points to the right of the red points in Figure 5.11a one can conclude that despite the vehicle having plenty of space to drive further away from the inner kerb of the roundabout, making the route easier by increasing the curvature, the agent correctly follows the given path.

Apart from the slight deviation from the centre of the lane at the bottom of the roundabout, the behaviour exhibited on this route is optimal and comparable to human behaviour.



(a) Route in context of the whole roundabout (b) Close up of route

Figure 5.12 Route followed in the 16m training roundabout scenario.

Figure 5.12 illustrates a training route from the 16m roundabout where the vehicle navigates through $360°$. In this case, if the vehicle were to follow the given path, the trailer would collide with the inner kerb of the roundabout since the rear trailer wheels follow a tighter radius path than the front tractor wheels. To avoid this, the agent must anticipate that a low radius section is approaching and move away from the centre of the lane before starting to turn. This is shown to occur in Figure 5.12b. After moving through the first $90°$ of the roundabout, the red path of the tractor starts to deviate from the given green route. Following this, to avoid a collision between the trailer and the kerb, the agent must continue to drive off-centre, allowing the trailer enough space to turn at a tighter radius. This behaviour is illustrated when the vehicle is moving from $90°$ to $270°$ from the starting position, where the red path of the tractor significantly dif-

fers from the green route. At $270°$ from the starting position, the tractor is now required to anticipate the right turn ahead to exit the roundabout. This implies that the optimal path would be for the vehicle to stay in the centre of the lane, since the trailer can follow a tight radius path to the right of the tractor unit without the risk of colliding with the kerb. This behaviour is shown in Figure 5.12b from $270°$ from the starting position till the end of the route, where the tractor unit is observed to move closer to the centre of the lane, until finally completing the route with the tractor-unit being at the centre of the lane. It should be noted that throughout the whole intersection, the trailer, shown by the magenta-coloured path, is maintained at the centre of the lane. This illustrates that the agent minimally deviates the tractor unit away from the centre of the lane such that the trailer does not collide with the inner kerb of the roundabout.

In Figure 5.12b one can also notice that the movement of the tractor unit, shown by the red path, is not smooth. In order to deviate from the centre of the lane, the tractor unit seems to repeatedly steer away from the roundabout, move forward, then steer into the roundabout, and move forward again. This is due to the small curvature of the roundabout, where the tractor has to frequently change the steering angle in order to stay closer to the centre of the lane. In such scenario, a more suitable behaviour would be to further increase the curvature of the path by deviating further away from the centre of the lane and therefore lowering the number of steering action changes required. Such rapid change in actions leads to suboptimal behaviour due to the jittering motion and lack of predictability in its movement.

Figure 5.12a illustrates the route in the context of the whole roundabout. When observing the middle section of the route, we can observe that the tractor's path is over the green waypoints representing the outer lane. This signifies that the tractor unit had to cross over into the outer lane in order to allow enough space to avoid a trailer collision. This is further observed in Figure 5.13 which illustrates a visual from the CARLA simulator while the tractor-trailer vehicle is halfway through this route. The tractor unit is observed to be on the outer lane while the trailer unit is on the inside lane, just avoiding the inner kerb. This is considered optimal behaviour since the tractor unit deviates as little as required. The route would be made easier if the tractor unit deviated further away from the centre of the lane, due to the higher curvature, but the negative reward, r_s, promotes the agent to stay as close to the centre of the lane as possible.

Evaluating the unsuccessful training routes, one can observe that 33% of the failures occur with the trailer colliding with the inner kerb of the roundabout. In these cases, the tractor unit fails to sufficiently deviate away from the centre of the lane to counteract for the tight radius path the trailer navigates through. The failing scenario would be similar to the one depicted in Figure 5.13, where the trailer would collide with the kerb. Furthermore, 38% of the failures occurred with the trailer colliding with the kerb while turning right, as shown in Figure 5.14. As discussed in Section 4.1.3, apart

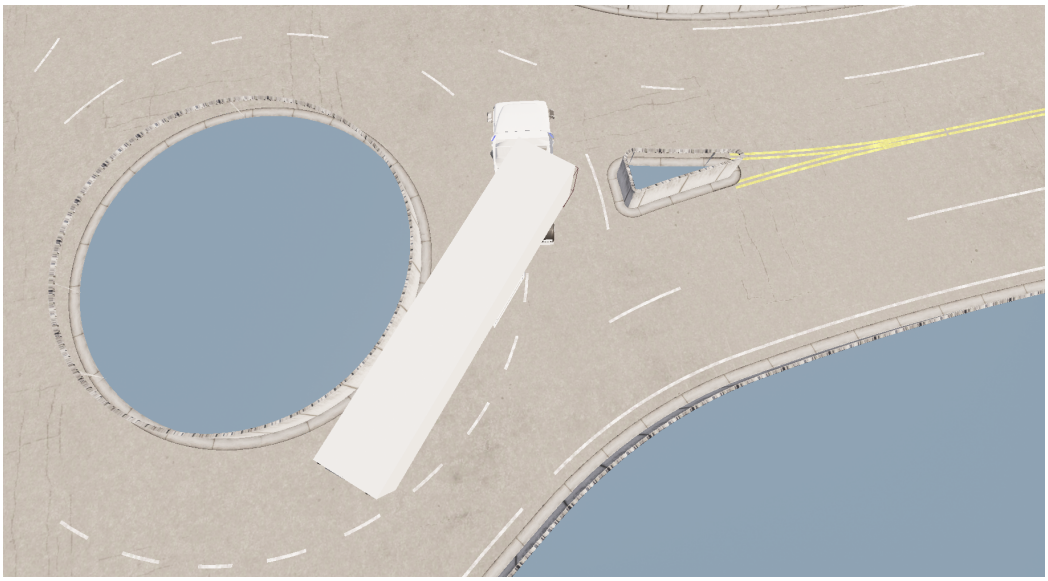


Figure 5.13 Tractor-trailer unit half-way through inferencing the route illustrated in Figure 5.12.

from varying the diameter of the roundabout, the entry-exit paths were designed with varying properties, differing in their ease of navigation. Figure 5.14 illustrates one of the more difficult entry-exit paths where the tractor-trailer vehicle has to deviate away from the centre of the lane in order to join the roundabout. A large amount of training examples required the vehicle to deviate to the right of the centre of the lane. On the other hand, examples which required deviation to the left of the centre of the lane, as required in Figure 5.14, were quite limited. The lack of suitable training scenarios for right-hand turns is the cause of the high collision rate while turning right.

The remaining 29% of collisions are due to the tractor unit colliding with the kerb. Most of these failures occur due to the tractor unit colliding with the splitter island, the small section of pavement in between the entry and exit lanes. Due to its small footprint on smaller diameter roundabouts, the distance sensors fail to capture such small obstacles well in advance. The remaining of these failures occur if the vehicle is initialised close to the kerb and the first action causes the vehicle to move closer to the kerb.

A similar trajectory analysis of the agent was performed using the testing scenarios. This tests whether the agent has learnt a general policy that can be used on different roundabouts with varying properties.

Figure 5.15 depicts the route navigated by the agent around the 40m testing roundabout scenario. Having a significantly large diameter, the roundabout allows the tractor-trailer vehicle to navigate through the intersection without requiring deviation from the centre of the lane. Such behaviour is observed by the agent as depicted in Figure 5.15b, where the red and magenta coloured paths, representing the tractor's and trailer's paths, are overlayed on top of each other above the green waypoints, which

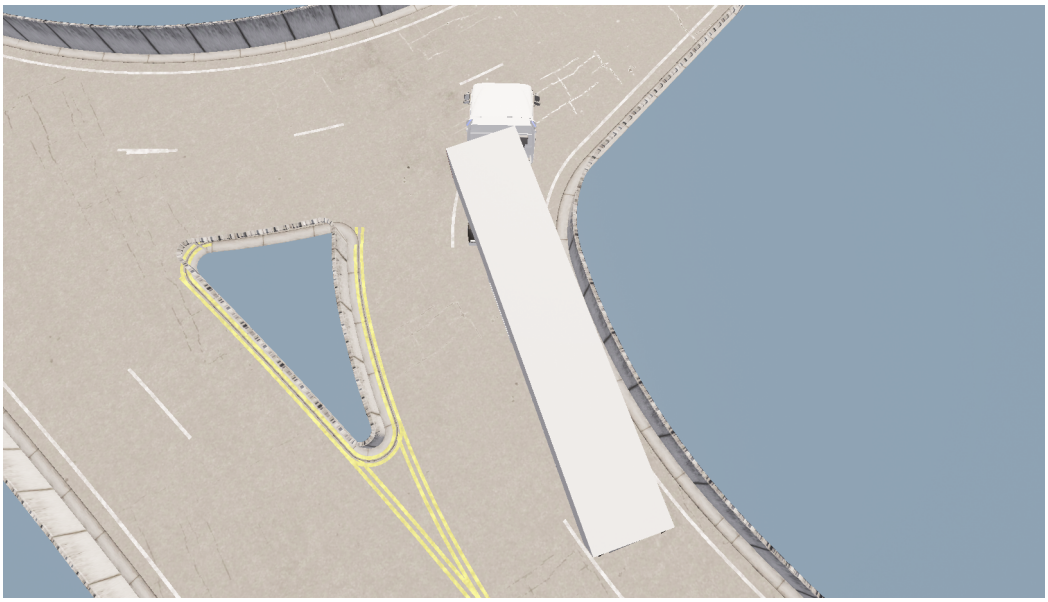
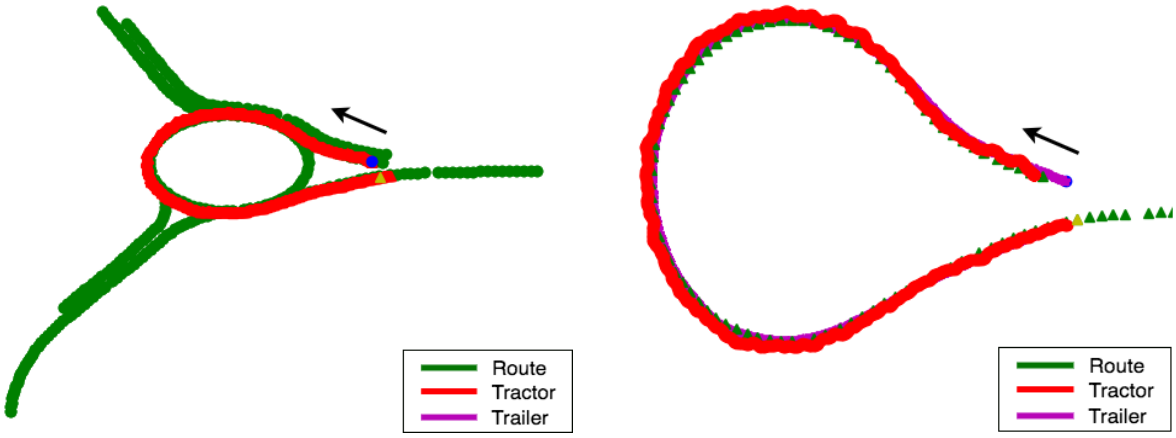


Figure 5.14 Trailer unit colliding with inside kerb during right turn on 50m roundabout.

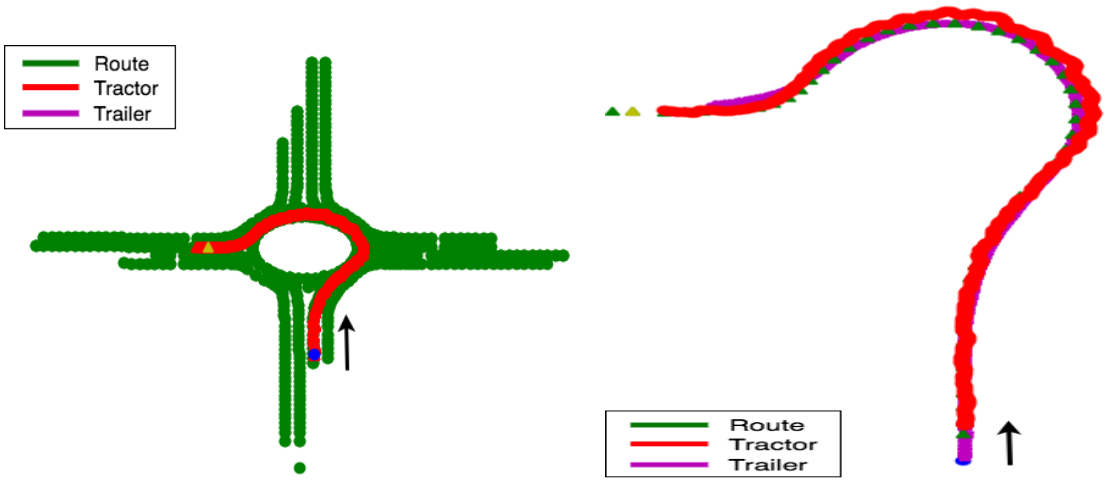


(a) Route in context of the whole roundabout (b) Close up of route

Figure 5.15 Route followed in the 40m testing roundabout scenario.

represent the tested route. Furthermore, on such a large curvature path, the agent is observed to navigate the vehicle smoothly.

Figure 5.15a displays the route in the context of the whole roundabout intersection. This enables us to observe that the vehicle followed the centre of the designated lane despite having the outer lane, which, if used, would have made the route easier due to the higher curvature.



(a) Route in context of the whole roundabout (b) Close up of route

Figure 5.16 Route followed in the 20m testing roundabout scenario.

Figure 5.16 illustrates the route followed by the agent when inferencing the model on the 20m testing roundabout. Congruent with the route developed in the training scenario, Figure 5.16b illustrates that the tractor-trailer vehicle stays at the centre of the lane up until it is necessary for the vehicle to deviate in order for the trailer not the collide with the inner kerb of the roundabout. While the tractor unit deviates from the path in the middle of the intersection, the trailer, illustrated by the magenta path, follows the centre of the lane. The trailer slightly deviates away from the centre of the lane while exiting the roundabout, since the tractor unit follows the centre of the lane at this section of the route rather than deviating further left in order to straighten the trailer. It should also be noted that the tractor units movement are relatively smooth when compared to the 16m training route shown in Figure 5.12b. This is due to the slightly larger roundabout diameter, which allows the vehicle to change its steering angle less frequently .

From Figure 5.16a, one can conclude that the agent only deviated the tractor unit as little as possible in order to successfully complete the route, exhibiting human-like behaviour. The green waypoints to the right of the red route signify that the agent may have deviated the tractor unit further away from the centre of the lane.

Comparing the behaviour and the routes followed by the agent in training scenarios, Figures 5.11 and 5.12, to the testing scenarios, Figures 5.15 and 5.16, one can observe very similar behaviour. Such similarity in the unseen testing scenarios indicates

that the model has learnt to generalise the navigation of roundabouts.

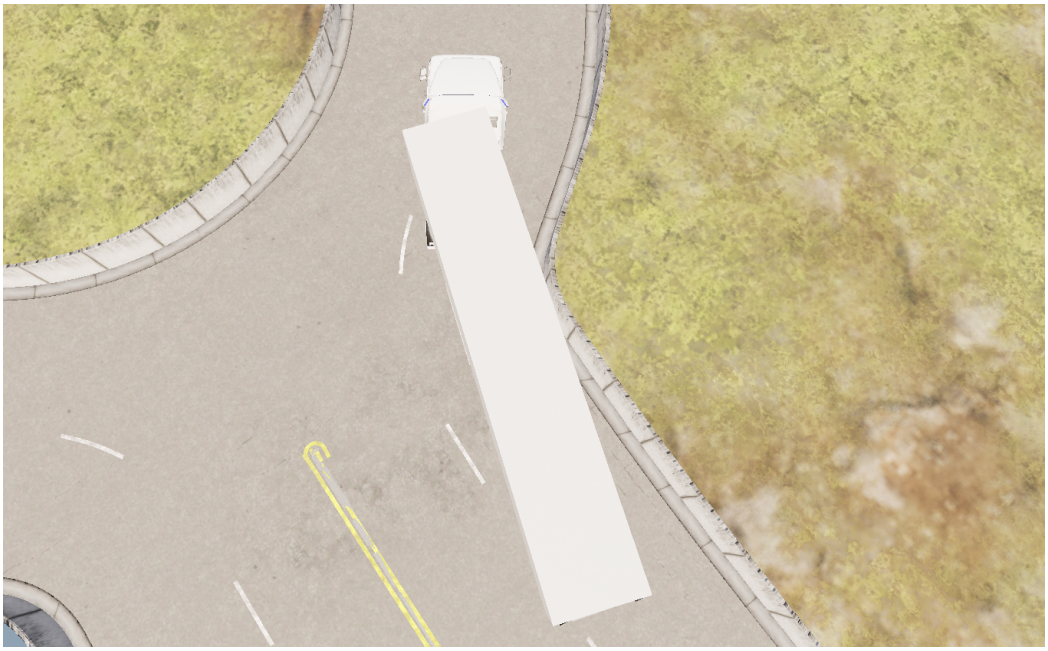


Figure 5.17 Trailer unit colliding with inside kerb during right turn on 20m roundabout.

Investigating the unsuccessful testing routes shows that, 75% of failures are related to routes in which a high deviation to the left of the centre of the lane, i.e. a low curvature right turn, is required. An example of this on the 20m testing roundabout is shown in Figure 5.17. The entry-exit points of the 20m roundabout were designed with a low curvature right turn, requiring the tractor-trailer vehicle to deviate left of the centre of the lane in order to make the turn. The remaining 25% of unsuccessful routes on the testing roundabout scenarios were either due to the trailer colliding with the inside kerb of the roundabout while the vehicle is turning left or due to the tractor unit colliding with the splitting lane. It should be noted that all routes of the 40m testing roundabout were successfully completed. Due to its relatively high diameter, the agent correctly anticipates all routes.

5.2.4 Comparison of results with current literature

To correctly evaluate the obtained results and gauge the performance of each model, one must compare our results to those obtained by the current state-of-the-art techniques used to solve this problem. Due to the limited research in this field, no such techniques were publicly available at the time of writing and therefore such methods cannot be applied to our dataset of roundabout intersections for accurate comparison. Due to time constraints, replication of RL based methods was not possible. Furthermore, rule-based methods include techniques outside the scope of our knowledge.

Despite this, our results will be compared to those obtained in current literature in order to evaluate the effectiveness of our framework. Previous research which laterally controls a tractor-trailer vehicle and does not include traffic, will be used to obtain a suitable comparison.

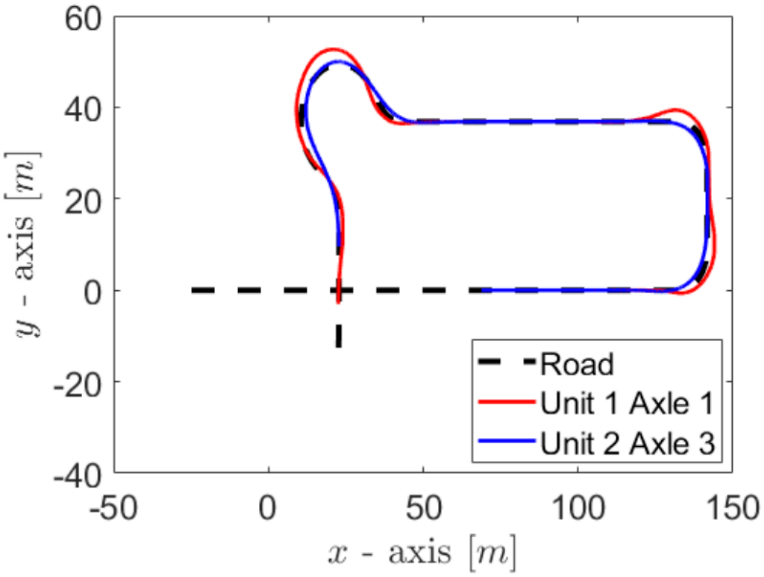


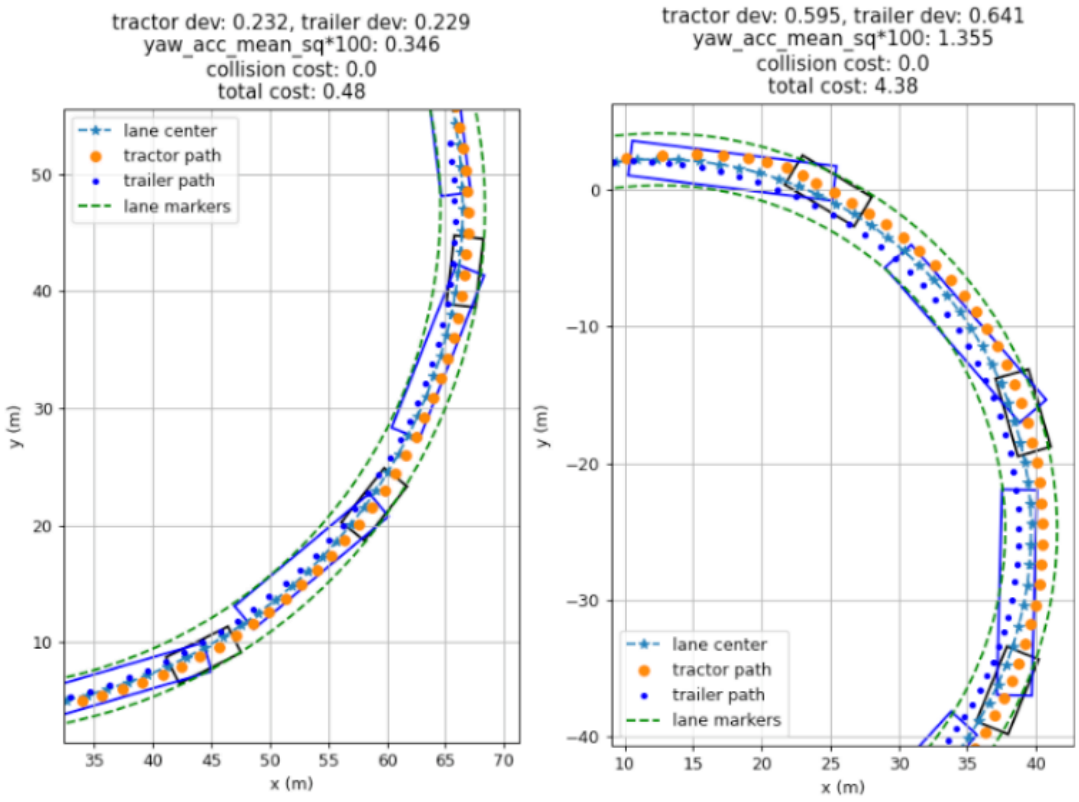
Figure 5.18 Road tracking of the tractor and trailer units observed in [14].

As discussed in Chapter 3, Diestra and Skouras [14] develop a number of trajectory planners using model predictive control. Aiming to maintain the trailer unit at the centre of the lane, they test their planners on a roundabout intersection where the vehicle has to navigate through $270°$ of the roundabout. Such roundabout was calculated to have a diameter of around 26.6m. Using the best trajectory planner proposed, the tractor unit deviates, at maximum, 4m away from the centre of the lane while the trailer unit deviates, at maximum, 2m away from the centre of the lane. Figure 5.18 illustrates the road tracking obtained where 'Unit 1 Axle 1' denotes the first axle of the tractor unit while 'Unit 2 Axle 3' denotes the third axle of the trailer unit.

Comparing the trajectory of our constant velocity action space model, depicted in Figure 5.16b, to that obtained by Diestra and Skouras in Figure 5.18, we can observe very similar behaviour by both models. In both figures, the tractor unit, marked in red, follows the centre of the lane until it is necessary for it to deviate away in order to avoid the trailer significantly veering away from the centre of the lane. For the route illustrated in Figure 5.16b, our models obtained a 0.69m and 0.38m lateral deviation for the tractor and trailer units, respectively. Such values are considerably lower than the 2m and 4m lateral deviations obtained by Diestra and Skouras's model, especially when considering that the roundabout used by Diestra and Skouras has a larger diameter.

As discussed in Chapter 3, the recent work of Zhang et al. [57] proposed a path-

planning approach for tractor-trailer vehicles using semi-supervised learning. The developed model is tested on artificially generated lanes. Figure 5.19 illustrates the trajectory analysis obtained from two testing scenarios. The roundabout in Figure 5.19a is calculated to have a diameter of 80m, with the tractor and trailer units experiencing a lateral deviation of 0.23m each. The roundabout in Figure 5.19b is observed to have a diameter of 50m. In this case, the tractor has a lateral deviation of 0.60m while the trailer experiences a lateral deviation of 0.64m.



(a) 80m diameter (b) 50m diameter

Figure 5.19 Trajectory analysis for the tractor-trailer vehicle [57].

Our constant velocity action space model obtained a tractor lateral deviation of 0.68m and a trailer lateral deviation 0.42m in a route from the 50m diameter roundabout. Comparing such values to those obtained by Zhang et al. [57] in the 50m diameter roundabout illustrated in Figure 5.19b, one can observe that our model obtained similar tractor lateral deviation. The difference of 0.08m in tractor lateral deviation can be considered as insignificant. A difference of 0.22m was observed in the trailer lateral deviation, where our model navigated the trailer closer to the centre of the lane.

The recent work of Ren et al. [59] proposes a double DQN RL environment to navigate a tracked agricultural tractor-trailer vehicle. As discussed in Chapter 3, training was performed using a simple path made up of two 45° angles and straight sections of road. The authors obtained a training success rate of 75%. Such success rate is slightly

lower than the success rate obtained by our constant velocity action space model, which obtained an 81% training success rate.

As one can observe, our constant velocity action space model obtains superior results when compared to the current literature. Apart from achieving a higher success rate of 81%, our model manages to navigate the tractor-trailer vehicle such that, on similar diameter roundabouts, the lateral deviation of the tractor and trailer units are less than those obtained in current literature.

5.2.5 Summary of Objective 2 Evaluation

The aim of this objective was to develop a suitable reinforcement learning environment for the lateral and longitudinal control of a tractor-trailer vehicle through a roundabout intersection. As mentioned, such work is the first of its kind and therefore limited knowledge about the state space and reward function which provide a functional model is known. This entails that multiple models using different input observations and reward functions were tested in order to obtain the functioning model observed above.

This section first illustrated the testing performed to find the optimal weight vector for the reward shaping function r_s. This weight vector varies the negative reward that is given to the agent based on the tractor's distance away from the centre of the lane. Testing four different weight vectors, a weight vector of $\frac{1}{400}$ was found to obtain superior results in both the training and testing scenarios. Other weight vectors were observed to overfit on the training roundabout intersection. The $\frac{1}{400}$ weight vector encouraged the agent to stay as close to the centre of the lane as possible while still allowing it to successfully complete most of the routes.

We then compared three different models which utilised different input observations. When compared to models solely using the angles between waypoints or the radii along the path, the 'Angles and Radii' model was observed to provide valuable information by which superior results were obtained, both in success rate and in distance to the centre of the lane.

A trajectory analysis of the PPO model using a weight vector of $\frac{1}{400}$ and both the angles to the waypoints and the radii along the path was also performed. The model was observed to maintain the vehicle close to the centre of the lane, only deviating when required. This behaviour was observed on both the training and testing roundabout, which ensured that the model did not overfit to the training scenarios. Despite mimicking human-like behaviour in terms of distance to the centre of the lane, the agent performed haphazard steering movements while circulating small-diameter roundabouts.

When comparing our results to those obtained by current literature, our constant velocity action space model achieves a higher success rate with smaller tractor and trailer lateral deviation when comparing on similar diameter roundabouts.

5.3 Objective 3: Comparing state-of-the-art RL algorithms

As discussed in Section 4.3, the DDDQN and SAC algorithms will be compared to the PPO algorithm. This allows for the comparison between the on-policy, policy gradient technique, PPO, the off-policy, q-learning technique, DDDQN, and the off-policy, actor-critic based technique, SAC. This enables us to evaluate the difference in behaviour between the three state-of-the-art RL algorithms.

It should be noted that apart from the different hyperparameters mentioned in Section 4.3, all three algorithms utilised the same state space, action space, and reward function. The algorithms utilise the best state space and reward function obtained in Sections 5.2.1 and 5.2.2. This entails that they utilise a weight vector of $\frac{1}{400}$ for the reward function r_s and also make use of both the angles to the waypoints and radii of the path as observations. This ensures an even playing field to be able to accurately compare the three algorithms.

Furthermore, identical testing roundabout scenarios and evaluation metrics utilised in Section 5.2 will be used in this section.

In order to correctly evaluate these models, the evaluation is made up of different components. This section will be divided as follows.

1. A quantitative evaluation including average training return graphs alongside evaluation metrics obtained by the models.
2. A qualitative evaluation assessing the unsuccessfully completed routes.
3. A qualitative evaluation assessing the successfully completed routes.
4. Summary of the findings

5.3.1 Quantitative Evaluation

Figures 5.20 and 5.21 illustrate the average training return obtained by the PPO, DDDQN, and SAC models, averaged over a 200 and 1500 episode sliding window, respectively. The 1500 episode sliding window smoothes out the average return, enabling us to obtain a more general view of how the average return differed.

Inspecting the returns obtained by the initial training episodes of all three algorithms, one can notice a significantly higher convergence rate for the PPO model when compared to the SAC and DDDQN models. While the PPO model managed to obtain an average return of 4 at around 1000 episodes, the SAC model required around 4500 while the DDDQN model required almost 6000 to reach the same return value. The difference in PPO and DDDQN convergence rates can be explained by the fact that the PPO model utilised a learning rate of 5×10^{-6} while the DDDQN model used a learning

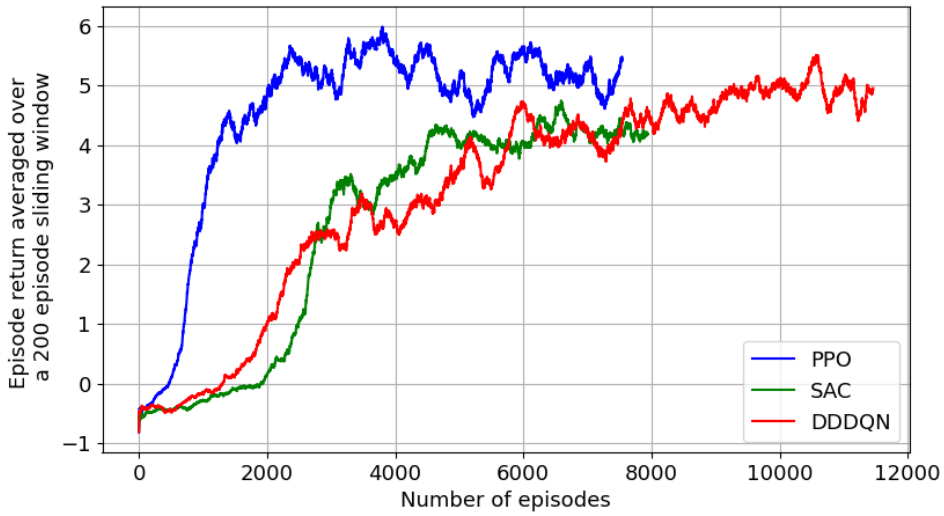


Figure 5.20 Average training return graph for the PPO, DDDQN, and SAC models, using a 200 episode sliding window.

rate of 5×10^{-7}. The lower learning rate utilised by the DDDQN model forces smaller changes in the action-value function in between iterations, resulting in more optimal convergence. This comes at the cost of time complexity since more training iterations are required to reach optimal convergence. A DDDQN model with a learning rate of 5×10^{-6} was also tested, but this failed to converge.

Since the SAC model used a learning rate of 5×10^{-5}, the slower convergence rate experienced by the SAC model cannot be attributed to the different learning rates. A possible reason for the slower convergence rate by the SAC model may be due to the fact that the SAC algorithm uses both policy gradient and an action-value function to learn an appropriate policy. Having both functions guiding the policy may inadvertently decrease the improvement the policy can make, leading to a slower convergence rate.

As can be observed in Figure 5.21, the PPO model manages to obtain the highest average return at around 5.1. The DDDQN model obtains a similar value of 5. Despite taking advantage of the benefits of both policy gradient and action-value based algorithms, the SAC model obtains the lowest average return of 4.2. It can also be noted that the PPO and SAC models require a similar amount of episodes to converge, around 7500 episodes. On the other hand, the DDDQN model, which makes use of the lowest learning rates, converges at around 10500 episodes.

Figure 5.22 illustrates the average training return graphs for the PPO, DDDQN, and SAC models, averaged over a 200 episode sliding window and in which the shaded area represents the 95% confidence interval. Similar variance is observed for all three models and such variance illustrates that each model has not fully converged yet.

The difference in the average return obtained can be further assessed by observ-

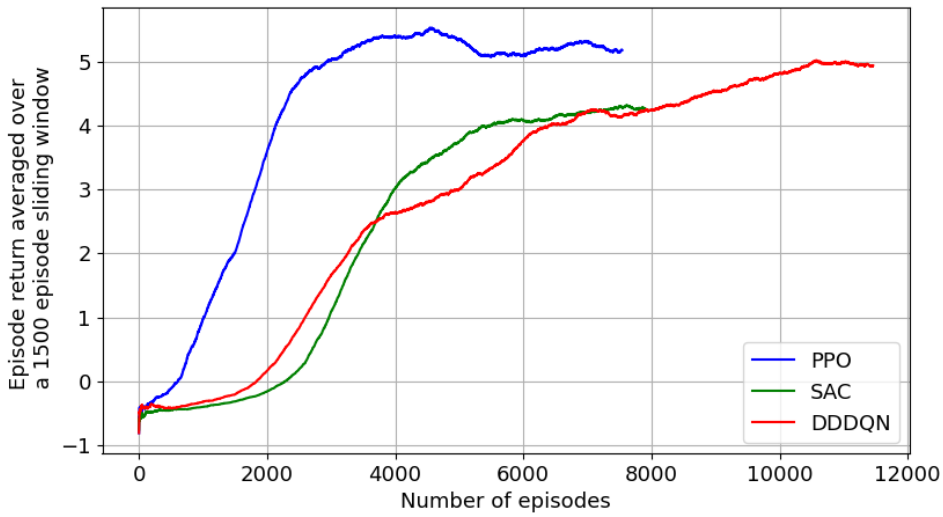
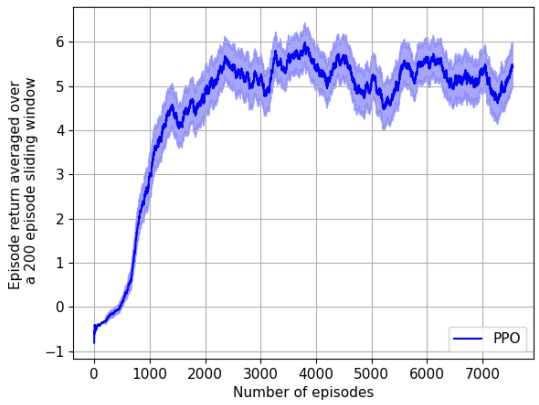
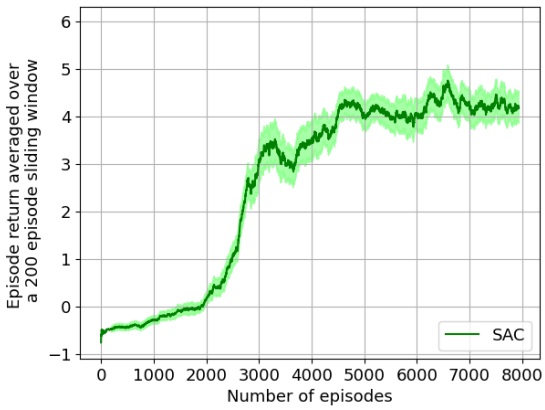


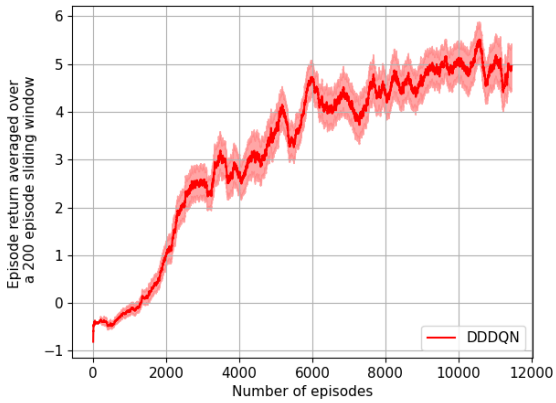
Figure 5.21 Average training return graph for the PPO, DDDQN, and SAC models, using a 1500 episode sliding window.



(a) PPO model



(b) SAC model



(c) DDDQN model

Figure 5.22 Average training return graphs for PPO, SAC, and DDDQN models over a 200 episode sliding window. The shaded area represents the 95% confidence interval.

Table 5.5 Evaluation metrics on training roundabout scenarios for the PPO, DDDQN, and SAC models

Training roundabout scenarios					
Model	**Success Rate**	**Tractor Collision Rate**	**Trailer Collision Rate**	d_c^t	d_c^{tt}
PPO	0.81	0.05	0.14	0.61	0.43
DDDQN	0.87	0	0.13	1.20	1.01
SAC	0.77	0	0.23	1.62	1.42

ing the training evaluation metrics illustrated in Table 5.5. As previously discussed the average return varies based on the distance of the tractor unit to the centre of the lane and whether the episode has been successfully completed or not. The PPO model obtains a success rate of 0.81 along with an average tractor unit distance to the centre of the lane, d_c^t, of 0.61m and an average trailer unit distance to the centre of the lane, d_c^{tt}, of 0.43m. The DDDQN model obtains a higher success rate of 0.87 when compared to the PPO model. Despite the higher success rate, the d_c^t and d_c^{tt} are computed at 1.20m and 1.01m, double the values obtained by the PPO model. Such high values of d_c^t and d_c^{tt} explain the lower average return despite a higher success rate obtained by the DDDQN model when compared to the PPO model. A similar trailer collision rate is obtained by the PPO and DDDQN algorithms, while a tractor collision rate of 0 is obtained by the DDDQN model.

The SAC model obtains a success rate of 0.77 and this is due to more trailer collisions when compared to the PPO and DDDQN models. Apart from the lower success rate, the lower average return obtained by the SAC model is due to the high values of d_c^t and d_c^{tt}. Being 1.62m and 1.42m respectively, they are almost triple the same values of the PPO model.

As mentioned, the distances d_c^t and d_c^{tt} are computed only using values from the successfully completed routes and are averaged over all the routes. In order to obtain a clearer picture of how the distances to the centre of the lane varied for each route, the difference in distances the d_c^t and d_c^{tt}, are computed and evaluated. This is performed between two models and for all the routes. Only the routes which have at least one successful run out of the 5 runs, in both models, will be included in these metrics.

It should be noted that such calculation was not computed when comparing different models in Sections 5.2.1 and 5.2.2 since such comparisons were focused on obtaining the highest success rate where the distances d_c^t and d_c^{tt} indicated the reason for the difference in success rates. In this case, a comparison between models using two different action spaces is being made and therefore apart from the success rate, the behaviour of the agent can be evaluated using the distances d_c^t and d_c^{tt}.

As shown in Table 5.6, when comparing between the PPO and DDDQN algorithm, the average difference in distance of the tractor unit to the centre of the lane for

Table 5.6 Training average differences in distances Δd_c^t and Δd_c^{tt}.

Models	$\overline{\Delta d_c^t}$	$\overline{\Delta d_c^{tt}}$
Between PPO and DDDQN	-0.56	-0.55
Between PPO and SAC	-0.91	-0.93

all training routes, $\overline{\Delta d_c^t}$, is $-0.56m$. The negative sign indicates that the PPO algorithm on average navigates the tractor unit 0.56m closer to the centre of the lane. The tractor unit has a width of 2.4m and the lane is 3.7m wide, therefore if the tractor unit was at the centre of the lane, a distance of 0.65m, calculated by $\frac{3.7}{2} - \frac{2.4}{2}$, separates the edge of the truck from the edge of the lane. We consider that if the difference in distance to the centre of the lane is greater than half the remaining space left, 0.33m, the vehicle's route is significantly different from one another. To put this value into context, a standard tractor unit tyre is 0.32m wide. This entails that we consider a route to significantly vary from another, if the tractor-trailer vehicle navigates, at maximum, a tyre's width difference away from the centre of the lane.

Therefore, the tractor unit in the DDDQN model being on average 0.56m further away from the centre of the lane, when compared to the PPO model, is considered significant. A more in-depth look reveals that 73% of training routes are more than 0.33m further away from the centre of the lane in the DDDQN model when compared to the PPO model. No routes were found to be significantly closer to the centre of the lane in the DDDQN model when compared to the PPO model. Similar findings are observed when calculating the difference in the trailer unit distance to the centre of the lane. Having a similar width to the tractor unit, the same cutoff point can be used. In 75% of the training routes, the trailer was found to be significantly further away from the centre of the lane in the DDDQN model when compared to the PPO model. Finally, one of the routes navigated by the DDDQN model was observed to deviate away from the centre of the lane, 1.2m and 1.1m more than the PPO model for the tractor and trailer unit's distances, respectively.

Comparing between the PPO and SAC model, $\overline{\Delta d_c^t} = -0.91m$ while $\overline{\Delta d_c^{tt}} = -0.93m$. This indicates that, on average, the PPO model navigates the tractor unit 0.91m closer to the centre of the lane and the trailer unit 0.93m closer to the centre of the lane. Taking 0.33m as the minimum difference in distance which considers a route significantly different from another, 95% of training routes are significantly navigated closer to the centre of the lane by the PPO model when compared to the SAC model. This is true for both the tractor and trailer unit distances. The remaining 5% of routes were found to be insignificantly different, while no routes were found to be navigated closer to the centre of the lane by the SAC model when compared to the PPO model. A minimum value of -2.14m indicates that, in a route successfully completed by both the PPO and SAC models, the tractor unit was navigated 2.14m closer to the centre of the lane by the

Table 5.7 Evaluation metrics on testing roundabout scenarios for the PPO, DDDQN, and SAC models

Testing roundabout scenarios					
Model	**Success Rate**	**Tractor Collision Rate**	**Trailer Collision Rate**	d_c^t	d_c^{tt}
PPO	0.77	0.02	0.22	0.58	0.40
DDDQN	0.79	0.01	0.20	1.22	1.05
SAC	0.87	0.00	0.13	1.60	1.34

PPO model when compared to the SAC model. A similar difference in trailer distance to the centre of the lane of -2.09m is found on this route.

Table 5.7 highlights the evaluation metrics obtained by the PPO, SAC and DDDQN models on the testing roundabout scenarios. The PPO model obtains a testing success rate of 0.77 along with the lowest values of d_c^t and d_c^{tt} at 0.58m and 0.40m respectively. The DDDQN model obtains an identical success rate despite the values d_c^t and d_c^{tt} almost being double those of the PPO model. Finally, the SAC model obtains a 0.87 success rate with 1.60m and 1.34m for d_c^t and d_c^{tt} respectively.

A possible explanation for the high testing success rate obtained by the SAC model is that the model was not suitable at capturing a suitable representation between the input states and the appropriate actions. This leads to a high bias and therefore generates an underfitting model. A suitable solution to combat underfitting models is by increasing the size and complexity of the internal neural network so that the model can learn a more complex relationship between the input states and the output actions. The SAC model presented in this section was trained using a q-learning neural network model and a policy neural network model composed of three hidden layers made up of 512, 512, and 1024 neurons, respectively. This is considered as a major increase in complexity to the standard neural network used, which is made up of two hidden layers composed of 256 neurons each. Models trained using the standard neural network configurations failed to converge to a usable model. A suitable SAC model which does not underfit can be developed by further increasing the complexity of the internal neural networks of both the q-learning and policy functions. Such hyperparameter tuning was not performed in this work due to time constraints and the computational complexity required to train each model. Despite this, a decrease in the severity of underfitting was noticed between models, which incrementally increased the size of the internal neural network, further confirming the requirement for a more complex neural network configuration.

When comparing between the PPO and DDDQN models on the testing scenarios, $\overline{\Delta d_c^t}$ is computed at -0.58m while $\overline{\Delta d_c^{tt}}$ is computed at -0.64m. Such values are presented in Table 5.8. This means that on average the PPO model manages to navigate the tractor unit 0.58m closer to the centre of the lane while the trailer unit is 0.64m closer to the

Table 5.8 Testing average differences in distances Δd_c^t and Δd_c^{tt}.

Models	$\overline{\Delta d_c^t}$	$\overline{\Delta d_c^{tt}}$
Between PPO and DDDQN	-0.58	-0.64
Between PPO and SAC	-0.85	-0.83

centre of the lane. Further analysis shows that for 81% of the routes, the PPO model navigated the vehicle closer to the centre of the lane when compared to the DDDQN model. No routes were found where the DDDQN algorithm performed significantly better lateral control of the vehicle.

Comparing between the PPO and SAC models, the following values are obtained $\overline{\Delta d_c^t} = -0.85m$ and $\overline{\Delta d_c^{tt}} = -0.83m$. When taking a difference of 0.33m as a cutpoint point for being a significant difference in lateral control, all routes where navigated significantly closer to the centre of the lane by the PPO model. Such results are similar to those obtained in the training scenarios.

5.3.2 Qualitative Evaluation - Unsuccessful Episodes

This section will focus on highlighting why and how different routes in different models failed to be successfully completed.

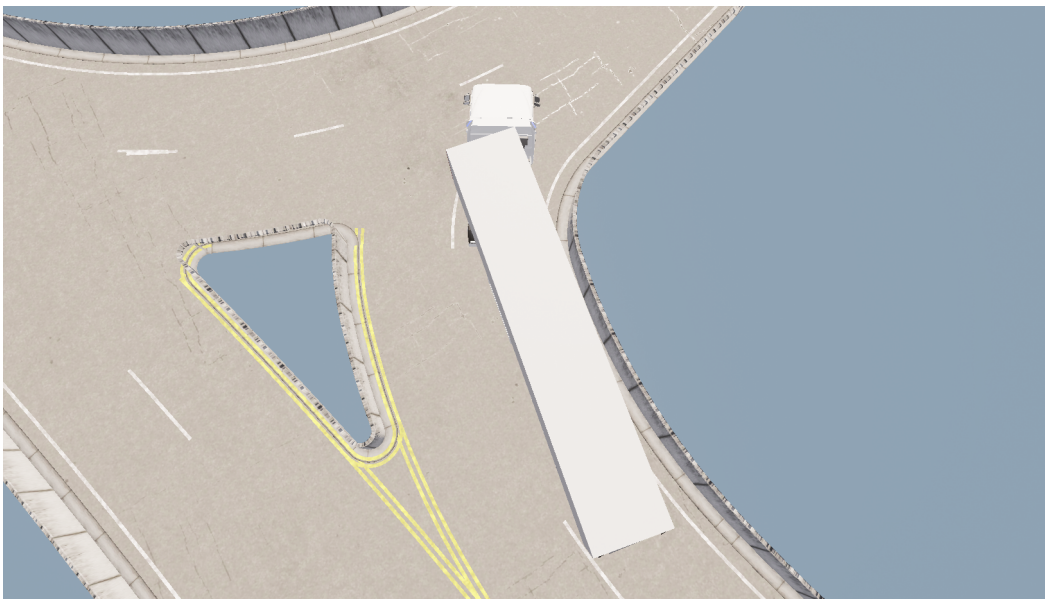


Figure 5.23 PPO model navigating the trailer through a low curvature entry-exit road.

In Section 5.2.3, Figure 5.23 illustrated that the PPO model failed to sufficiently navigate away from the centre of the lane in order to pull out of a low curvature entry-exit road. This route was successfully completed by both the DDDQN and SAC models. Figure 5.24 illustrates how the DDDQN model managed to navigate through the low curvature entry-exit road with the SAC model performing similarly. Anticipating the tight

curvature, the DDDQN model steers the tractor unit away from the kerb well before the turn itself. This allows the tractor-trailer vehicle to navigate a higher radius curvature, which is required in order to make the turn. While in Figure 5.23 the tractor and trailer units are both inside their appropriate lanes, albeit at its edge, the tractor and trailer units in 5.24 are well into the neighbouring lane. Although one would prefer if the vehicle stayed within its lane, the curvature of this entry-exit road forces the vehicle to cross into other lanes in order to make the turn. Furthermore, it should be noted that in order to avoid collisions while navigating this entry-exit road, the DDDQN model also has to cross over into the neighbouring lane for the first few meters after joining the roundabout. This allows the trailer to clear the tight curvature.

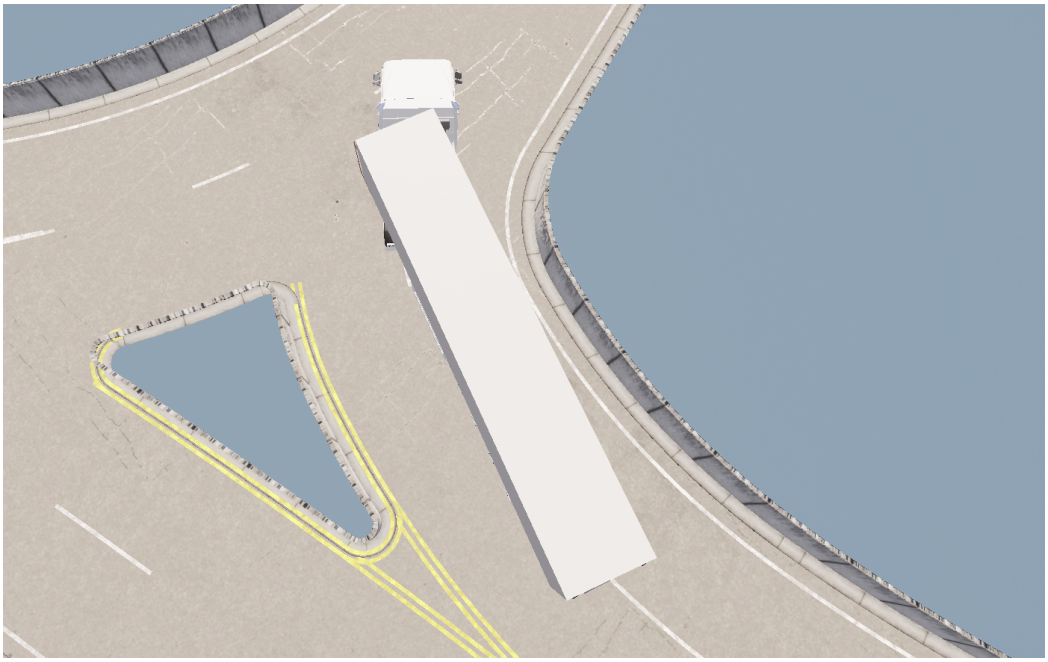


Figure 5.24 DDDQN model navigating the trailer through a low curvature entry-exit road.

The successful completion of this route comes at a cost of a significantly high distance to the centre of the lane. This is visually observed by the tractor-trailer vehicle being navigated at the centre of both lanes for the remaining part of the route. The remaining part of the route does not require the tractor-trailer vehicle to deviate away from the centre of the lane. For this specific route, values $d_c^t = 2.08m$ and $d_c^{tt} = 1.61m$ were obtained by the DDDQN model.

Despite the SAC model having the highest value of d_c^t and d_c^{tt} from all three models, the SAC agent fails to navigate a route on the 20m testing roundabout which requires the vehicle to deviate away from the centre of the lane. Furthermore, this specific route is successfully completed by both the PPO and DDDQN models. A visual illustration of where the trailer collides with the kerb is shown in Figure 5.25.

In order to better understand why the agent failed to successfully navigate this

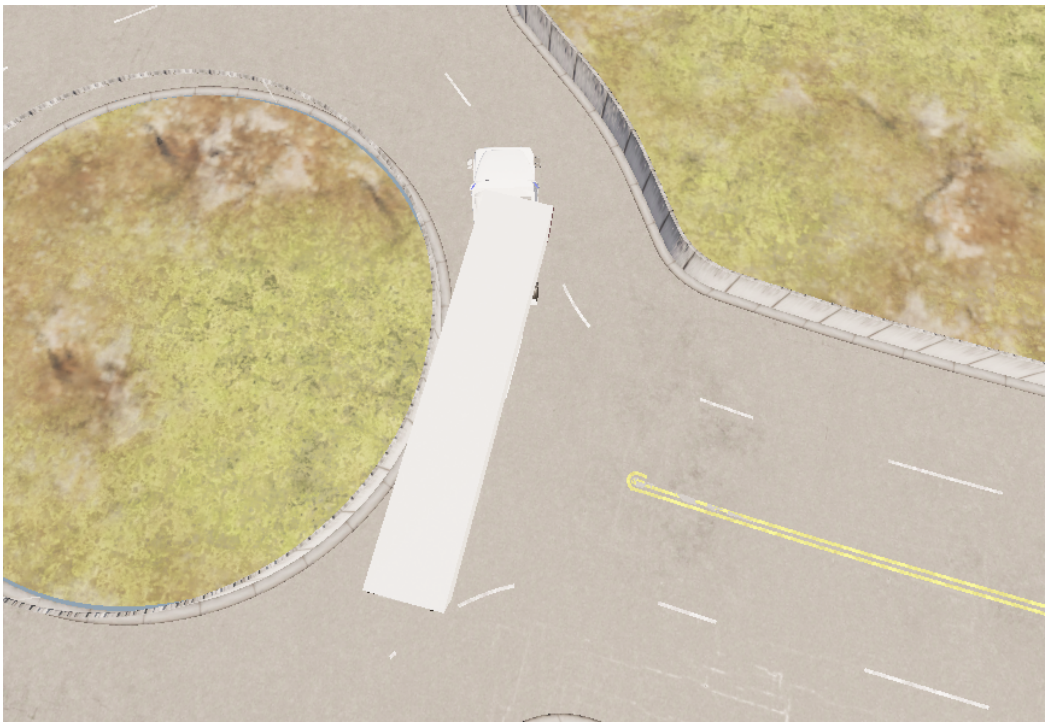


Figure 5.25 SAC model colliding with insider kerb of roundabout on 20m testing roundabout.

route, the path taken by the PPO agent will be compared to that taken by the SAC agent. Figure 5.26b illustrates the path navigated by the PPO agent to successfully complete the route while Figure 5.26a showcases the path taken by the SAC agent which terminated with the trailer colliding with the roundabout as shown in Figure 5.24.

As illustrated in Section 5.2.3, the PPO model maintains the tractor-trailer vehicle close to the centre of the lane throughout the whole route, only deviating when required to avoid collisions. On the other hand, the SAC model significantly deviates away from the centre of the lane at the very beginning of the route where, as shown by the PPO path, such deviation is not required. Despite this deviation at the beginning of the route, the SAC model maintains the tractor unit at the centre of the lane where, as shown by the PPO path, the tractor unit needs to be steered right in order to avoid a trailer collision.

The SAC model seems to follow different behaviours on different routes. It manages to navigate through a low curvature entry-exit path like the one depicted in Figure 5.24 while it fails to sufficiently navigate away from the centre of the lane in a more simple scenario depicted in Figure 5.26a.

A possible explanation for the SAC model failing to successfully complete the route illustrated in 5.26a is that the model tried to average out the whole route. The SAC model is looking at the bigger picture of the route, and not just the coming waypoints, navigating the agent in the general direction of future waypoints. Internally, this would be represented by the neural network giving a higher weighting to the state inputs which provide information about distant waypoints. On the other hand, the PPO

(a) SAC Model (b) PPO Model

Figure 5.26 Route from 20m testing roundabout which the SAC model fails to complete.

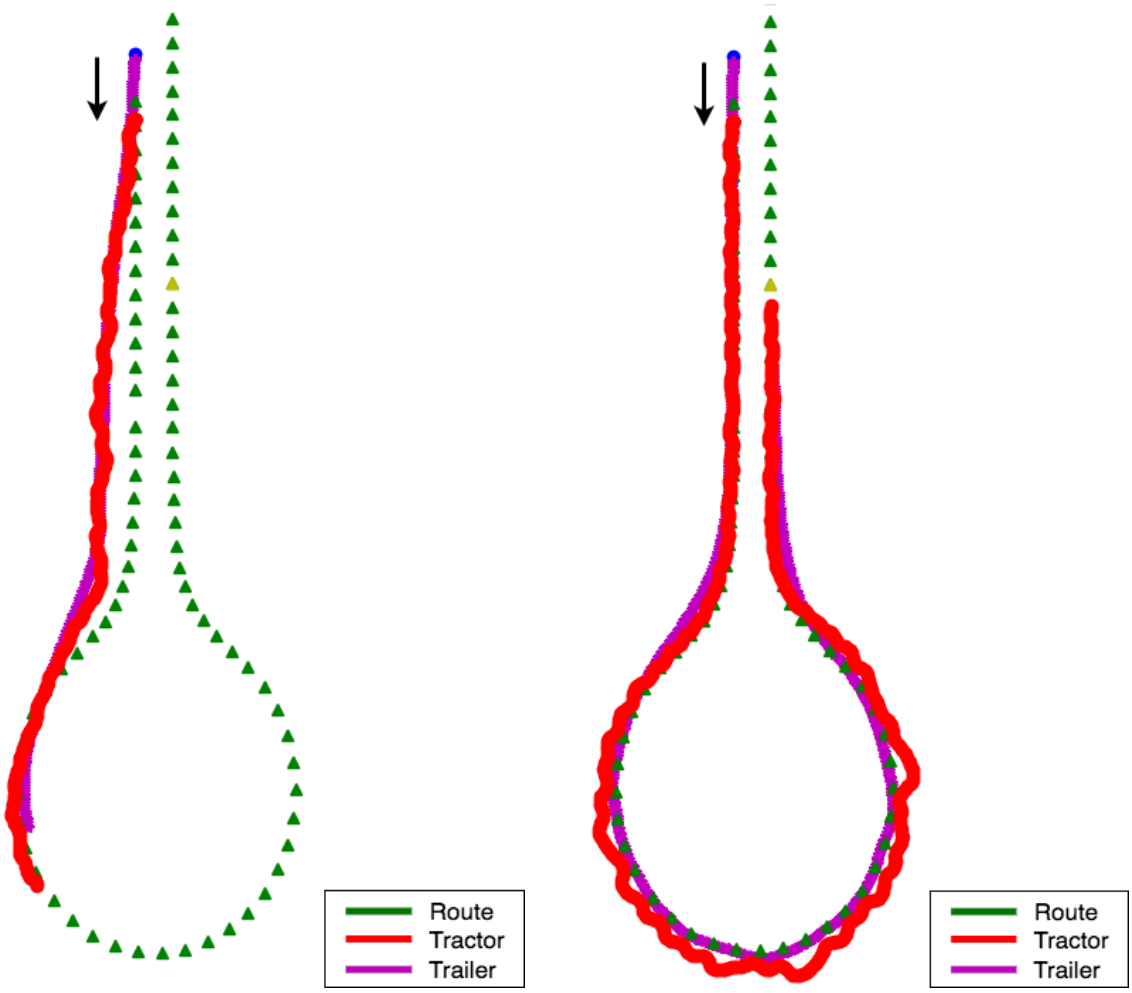
model internally gives more importance to state inputs, which provide information about waypoints close to the agent, while utilising the information about distant waypoints to better position the vehicle for upcoming curvatures. Keeping this in mind, a possible explanation to why the SAC model is able to successfully navigate through the tight curvature entry-exit road depicted in Figure 5.24, is that the curvature despite having a lower radius, is spread out over a larger number of waypoints. This means that distant waypoints, which are given higher importance by the SAC model, are also indicating a tight radius turn.

The input states have a wide range of waypoints at different distances away from the current waypoint, ranging from 1 to 12 waypoints ahead. Due to the small diameter of the roundabout visualised in Figure 5.26a, if the tractor unit is at the entry of the roundabout, 12 waypoints ahead may already be close to the desired roundabout exit and such waypoint would have a heading similar to the current one. This gives a false impression to the model that the route is relatively straight. On the other hand, considering Figure 5.24, if the tractor unit is starting to approach the entry-exit road, the heading of the waypoint, 12 waypoint ahead, is significantly different to that of the current waypoint, indicating an upcoming tight curve. This is due to the large curvature of the roundabout, where 12 waypoints ahead would mean that the vehicle has only navigated through a small portion of the roundabout.

Figure 5.27 illustrates a route which the DDDQN model fails to successfully complete while the PPO and SAC models successfully complete. The DDDQN model fails in a similar manner to that showcased in Figure 5.25. As one can observe in Figure 5.27a, the DDDQN agent deviates away from the centre of the lane as soon as the vehicle starts moving forward. As shown by the PPO model in Figure 5.27b, this is not required. Furthermore, in the initial straight section of the route, the tractor unit, depicted by the red path, is observed to laterally move left and right in the DDDQN model. Similar behaviour is observed by the PPO model, but this is to a much lesser degree.

At the point when the trailer unit collides with the kerb in the DDDQN model, the tractor unit is observed to be at the centre of the lane. When the tractor-trailer vehicle is at the same location while being navigated by the PPO model, the tractor unit, shown by the red path, is much further away from the centre of the lane, shown by the green path. At this point, the trailer unit, show by the magenta path, is at the centre of the lane. Despite the significantly high values of d_c^t and d_c^{tt} for the DDDQN model, the agent failed to deviate away from the centre of the lane when required to do so.

In the same roundabout and from the same starting point depicted in Figure 5.27, the DDDQN agent also performed routes exiting at $180°$ and $270°$ from the starting point. These routes were successfully performed with a 100% success rate. The route which exits after $270°$ from the starting point is illustrated in Figure 5.28. When looking at the same point in Figure 5.28, at which the vehicle collided with the kerb in Figure 5.27a, one



(a) DDDQN Model (b) PPO Model

Figure 5.27 Route from 20m testing roundabout which the DDDQN model fails to complete.

can see very minimal difference in the tractor's location. Despite this, the tractor unit in Figure 5.28 is observed to move away from the centre of the lane a few waypoints earlier than the tractor in Figure 5.27a. This allows for the tractor unit to be able to navigate a larger curvature path, avoiding a collision between the trailer and the kerb. This also allows the trailer unit to be further away from the roundabout in Figure 5.28 when compared to that in Figure 5.27a.

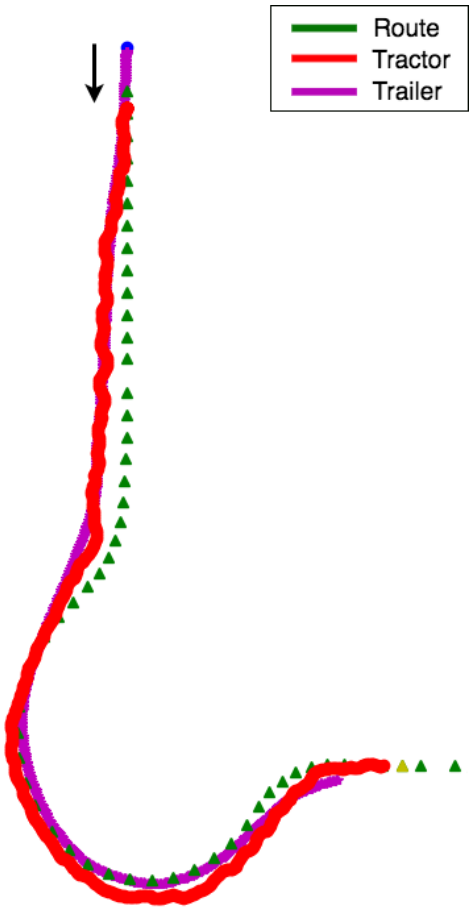


Figure 5.28 DDDQN model completing a route on 20m roundabout with the same starting point as Figure 5.27a.

The difference in behaviour by the DDDQN model that allows it to successfully complete the route in Figure 5.28 while failing the route in Figure 5.27a can be explained by the nature of the route itself. Similar to the SAC model, the DDDQN is observed to give more importance to information obtained from waypoints which are further away from the current waypoint. In Figure 5.27a the heading of a waypoint, 12 waypoints ahead, may be similar to that of the current waypoint, albeit in the opposite direction. On the other hand, in the route depicted in Figure 5.28, the heading of the waypoint, 12 waypoints ahead, from the point at which the trailer collided in Figure 5.27a, would possibly have a significantly different heading. This is because the agent's expected exit is much closer and at a different angle to that in Figure 5.27a. Such a difference in heading would allow the agent to correctly anticipate a tight radius turn.

5.3.3 Qualitative Evaluation - Successful Episodes

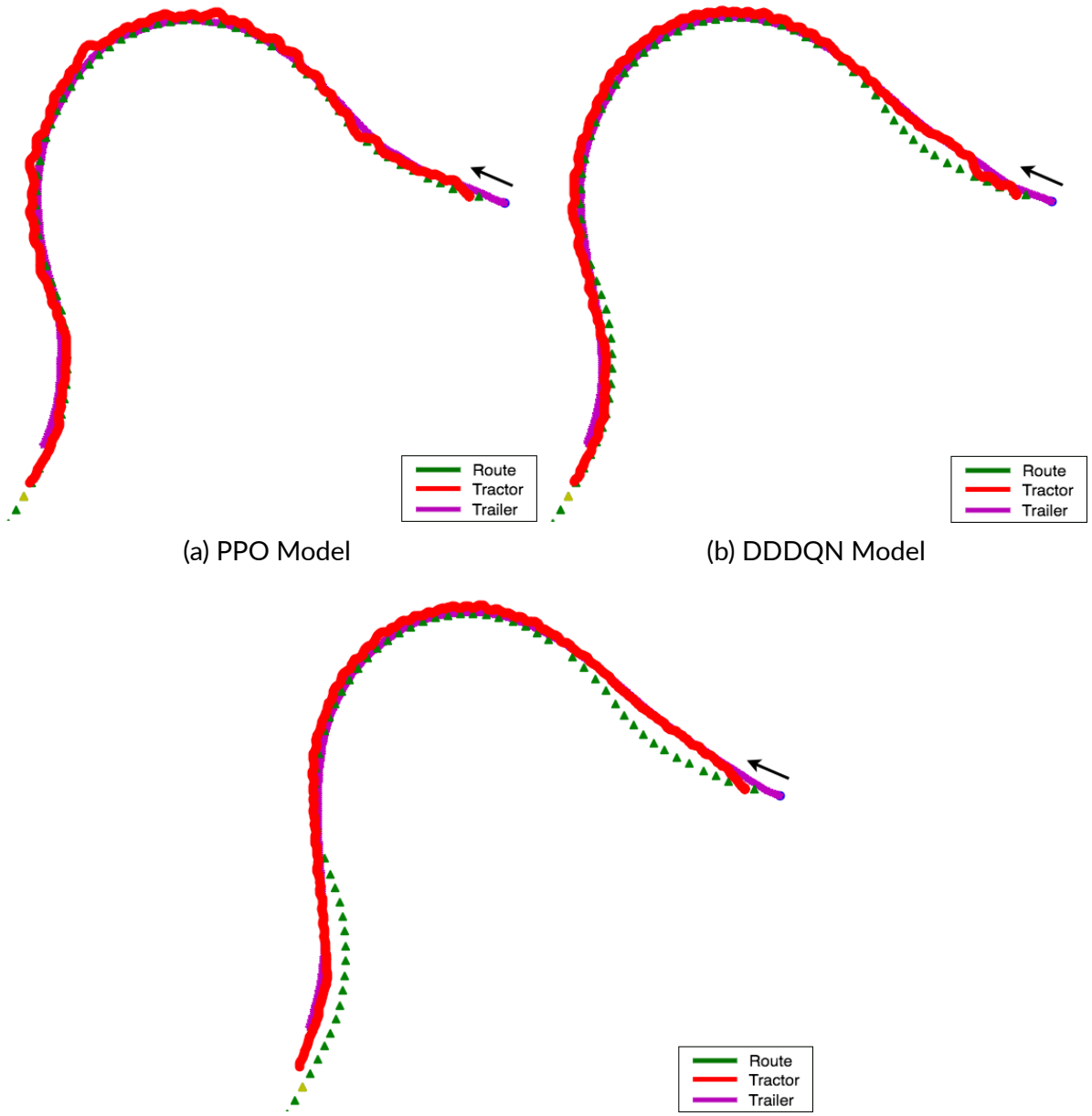
This section will focus on assessing the different behaviours exhibited by each RL model using routes which all three models managed to successfully complete.

Figure 5.29 illustrates a route from the 40m testing roundabout which all three models successfully managed to complete. The PPO agent, shown in Figure 5.29a, manages to navigate the vehicle close to the centre of the lane during the whole route. The PPO agent obtained a d_c^t of 0.58m and a d_c^{tt} of 0.37m. On the other hand, the SAC model, shown in Figure 5.29c, only managed to maintain the vehicle at the centre of the lane while circulating around the roundabout. Before joining and after exiting the roundabout, the agent is observed to take the shortest path, a straight line to the waypoint. It should be noted that the reward function only encourages the agent to maintain the tractor unit close to the centre of the lane and to complete the route. The agent is not rewarded for completing the route in the shortest timeframe. The SAC agent obtained a d_c^t of 1.32m and a d_c^{tt} of 1.08m. The DDDQN model, shown in Figure 5.29b, generates a path which is in-between the two extremes. The initial part of the route is observed to start off with the vehicle close to the centre of the lane, but after a few waypoints, the straight line route is chosen. Despite this, after exiting the roundabout, the DDDQN model is observed to navigate the vehicle much closer to the centre of the lane when compared to the SAC model. The vehicle is still navigated further away from the centre of the lane compared to the PPO model. The DDDQN agent obtained a d_c^t of 0.94m and a d_c^{tt} of 0.69m. The values of d_c^t and d_c^{tt} for each model are congruent with the visual observations.

From Figure 5.29, one can observe that the PPO model, despite obtaining the lowest distance to the centre of the lane, generated a jagged path. This is due to the fact that the PPO model significantly varies the steering angle of the tractor unit from one timestep to another and therefore an uneven path where the tractor unit continually moves left and right is obtained. Such behaviour is not observed in the DDDQN or SAC models, where the path of the tractor unit is significantly smoother.

The PPO model may be moving the vehicle laterally in order to slow the vehicle down and allow the model to maintain the least possible distance away from the centre of the lane. Despite this possibly being true, the velocity of the vehicle was not changed since the DDDQN and SAC models are able to smoothly navigate the vehicle as close to the centre of the lane as the PPO model around the curvature of the roundabout.

Figure 5.30 depicts a route from the 20m testing roundabout which all three models managed to successfully complete. Similar to the previous example, the PPO model navigated the vehicle closest to the centre of the lane, obtaining a d_c^t of 0.65m and a d_c^{tt} of 0.35m. The DDDQN model manages to obtain higher values of d_c^t and d_c^{tt}, computed at 1.36m and 1.22m. These significantly higher values for the distance of the tractor and



(a) PPO Model

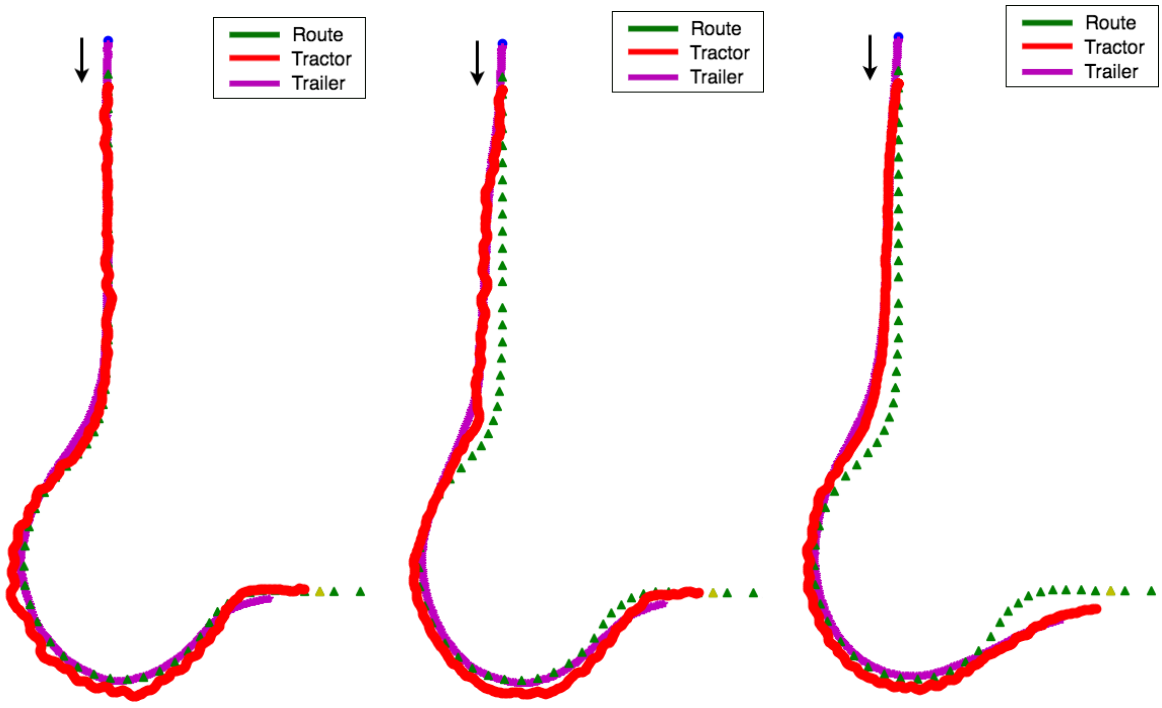
(b) DDDQN Model

(c) SAC Model

Figure 5.29 Route from the 40m roundabout which the PPO, DDDQN, and SAC models managed to complete.

trailer unit away from the centre of the lane are due to the vehicle deviating away from the centre of the lane in the initial part of the route and after exiting the roundabout. While turning around the roundabout, a similar deviation away from the centre of the lane can be observed. Similar to the previous example, the SAC model navigates the vehicle close the centre of the lane while turning around the roundabout, while when entering or exiting the roundabout, a high deviation from the centre of the lane can be observed. The SAC model obtained a d_c^t of 1.46m and a d_c^{tt} of 1.15m. Such values are very similar to those obtained by the DDDQN model. This further confirms that the major difference in deviation from the centre of the lane is obtained while entering or exiting the roundabout. The route in Figure 5.29 has a large section devoted to entering and exiting the roundabout when compared to the route in Figure 5.30. This allows the vehicle to gain larger values of d_c^t and d_c^{tt}.

Once again, the SAC model produces the smoothest path, whereas the PPO model is observed to produce a jagged path with the steering angle being continually changed.



(a) PPO Model (b) DDDQN Model (c) SAC Model

Figure 5.30 Route from the 20m roundabout which the PPO, DDDQN, and SAC models managed to complete.

5.3.4 Comparison of results with current literature

As discussed in Section 5.2.4, the PPO model navigates the tractor-trailer vehicle with similar behaviour to that observed in the work of Diestra and Skouras [14]. While achieving lower tractor and trailer lateral deviation values, the PPO model also achieves a higher success rate.

Our DDDQN and SAC models obtain 87% and 77% training success rates, respectively. When comparing to the 75% success rate obtained by Ren et al. [59], our DDDQN model is observed to obtain superior results while our SAC model obtains comparable results.

Comparing the trajectory analysis of the DDDQN and SAC models in Figure 5.30 to that obtained by Diestra and Skouras [14] in Figure 5.18, one can observe some differences. Despite circulating the roundabout in a similar manner, where the trailer lateral deviation is kept to a minimum, the tractor unit in both of our DDDQN and SAC models deviates away from the centre of the lane before joining and after exiting the roundabout. The tractor and trailer lateral deviations for the DDDQN model, when applied to the route illustrated in Figure 5.30b, are 1.36m and 1.22m, respectively. 1.46m and 1.15m tractor and trailer lateral deviations are observed by the SAC model when navigating through the route illustrated in Figure 5.30c. Despite obtaining higher values than our PPO model, such values are significantly lower than the tractor and trailer lateral deviation values of 4m and 2m, obtained by Diestra and Skouras.

Table 5.9 Tractor and trailer lateral deviations obtained by the PPO, DDDQN, and SAC models on the route depicted in Figure 5.29

Models	d_c^t	d_c^{tt}
PPO	0.58	0.37
DDDQN	0.94	0.69
SAC	1.32	1.08

Figure 5.29 illustrates the trajectory analysis of our PPO, DDDQN, and SAC models on a 40m diameter roundabout while Table 5.9 illustrates the tractor and trailer lateral deviation values obtained by our models on this route. Comparing these deviation values to the 0.60m and 0.65m tractor and trailer deviation values obtained by Zhang et al. [57], one can observe that our DDDQN and SAC models achieve higher lateral deviation values. Despite this, visually analysing the path followed in Figure 5.29 illustrates that when circulating the roundabout, both the tractor and trailer units are close to the centre of the lane. Such high average deviation values are due to the tractor unit moving away from the centre of the lane before joining and after exiting the roundabout.

5.3.5 Summary of Objective 3 Evaluation

This objective was aimed at finding the optimal RL algorithm to solve the problem at hand while also examining the difference in behaviour between different RL algorithms. The RL algorithms varied from the on-policy, policy gradient based algorithm, PPO, the off-policy, q-learning based algorithm, DDDQN, and the off-policy, actor-critic based algorithm, SAC.

Initially, a quantitative evaluation of the average return and different evaluation metrics was performed. This illustrated the superiority of using the PPO algorithm for this problem, achieving the highest average return during training. Furthermore, the high return value is complemented by a high success rate and the lowest values for the distances of the tractor and trailer units to the centre of the lane. The DDDQN model obtained slightly higher success rates than the PPO model at the cost of double the distances d_c^t and d_c^{tt} when compared to PPO. The SAC model was observed to obtain a higher testing success rate when compared to its training success rate. This led to the conclusion that the model failed to develop a suitable relationship between the input states and the output actions, underfitting, due to the size and complexity of the neural networks which represent the q-learning function and the policy gradient function. Additionally, the SAC model obtained the highest values of d_c^t and d_c^{tt}, being triple the values obtained by the PPO model.

Performing a qualitative analysis of each model indicated that due to their larger values of d_c^t and d_c^{tt}, the DDDQN and SAC models manage to successfully complete a route which includes a low curvature entry-exit road and therefore required significant deviation away from the centre of the lane. Such a route was consistently unsuccessfully completed by the PPO algorithm.

Further analysis showed the limitations of the DDDQN and SAC models. They significantly deviate away from the centre of the lane on straight sections of the road, found before entering and after exiting the roundabouts. Despite not encouraging the agent to do so, these two models seem to find the shortest route, a straight-line path, whenever possible. This behaviour is more prevalent in the SAC model. Furthermore, after examining some unsuccessfully completed routes, both the DDDQN and SAC models were noticed to give more importance to input states which give information about the waypoints furthest away from the agent.

Despite obtaining the least value for the distance to the centre of the lane, the PPO model produces jagged vehicle movement. The tractor-trailer vehicle is observed to continually change its steering angles. Such behaviour is not observed in the SAC model, with the DDDQN model being a balance of the two. Although such movement is not considered optimal behaviour, the significantly lower values of d_c^t and d_c^{tt} far outweigh the loss of smooth movement. This makes the PPO algorithm the most optimal model

for our autonomous driving problem.

Our PPO, DDDQN, and SAC models achieve comparable or higher success rates when compared to current literature. In terms of lateral deviation, our PPO model manages to navigate the tractor-trailer vehicle close to the centre of the lane. Our DDDQN and SAC models obtain higher lateral deviation values than current literature, but such high values occur before entering and after exiting the roundabout.

5.4 Objective 4: Incorporating acceleration actions

The previously evaluated models made use of the constant velocity action space, where the velocity was maintained at a constant value of $8km/h$ using a PID controller. In order to incorporate acceleration actions and enable the PPO algorithm to control both the longitudinal and lateral movements of the vehicle, the variable velocity action space, defined in Section 4.4 was used, allowing the agent to vary its velocity by providing actions with different acceleration values.

The evaluation of the addition of the acceleration actions to the policy optimization algorithm is divided into two sections,

1. Development of the variable velocity action space model.
2. Comparing the constant velocity and variable velocity action spaces.

5.4.1 Development of the variable velocity action space model

As described in Section 4.4, a negative reward based on the difference in velocities from the previous to the current timestep is given to the agent. This encourages a more stable velocity throughout the episode. In order to justify and demonstrate the benefits of adding such a reward, a model using the variable velocity action space without the velocity smoothing reward component (referred to as 'variable.velocity.no.smoothing'), was evaluated. This model was compared to a model using the velocity smoothing reward component (referred to as 'variable.velocity.smoothing'). It should be noted that these two models were trained using identical hyperparameters and only differed in the action space used.

Figure 5.31 illustrates the velocity of the agent while completing a route in the 40m testing roundabout. An unseen testing roundabout was utilised to observe the generalisability of the model and ensure that such behaviour is not limited to the roundabouts it was trained on. Two velocity graphs are plotted, the blue plot uses the 'variable.velocity.no.smoothing' model, while the red plot uses the 'variable.velocity.smoothing' model. Both velocity graphs were obtained by performing the same routes.

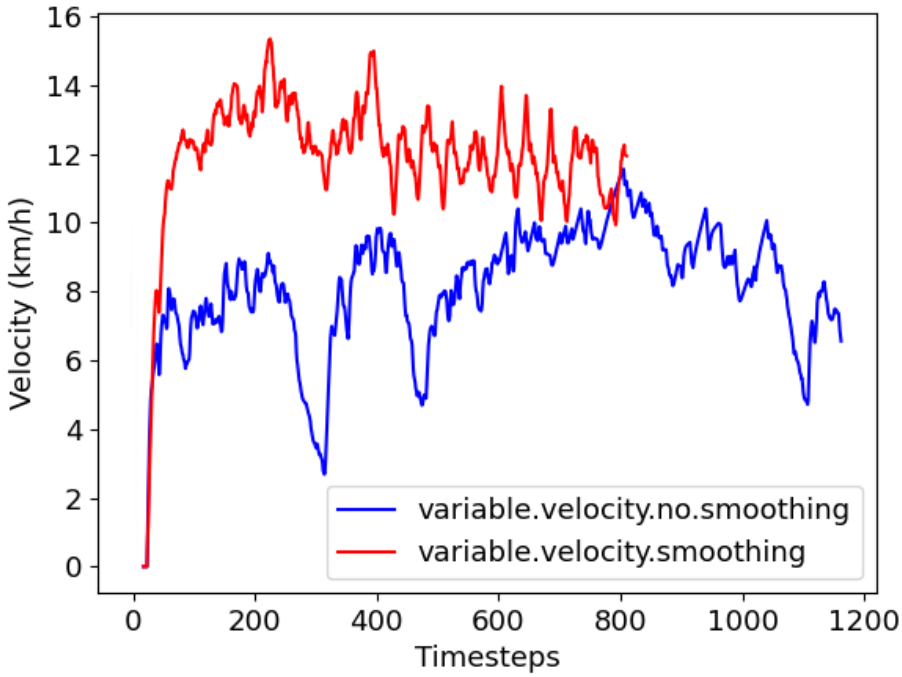
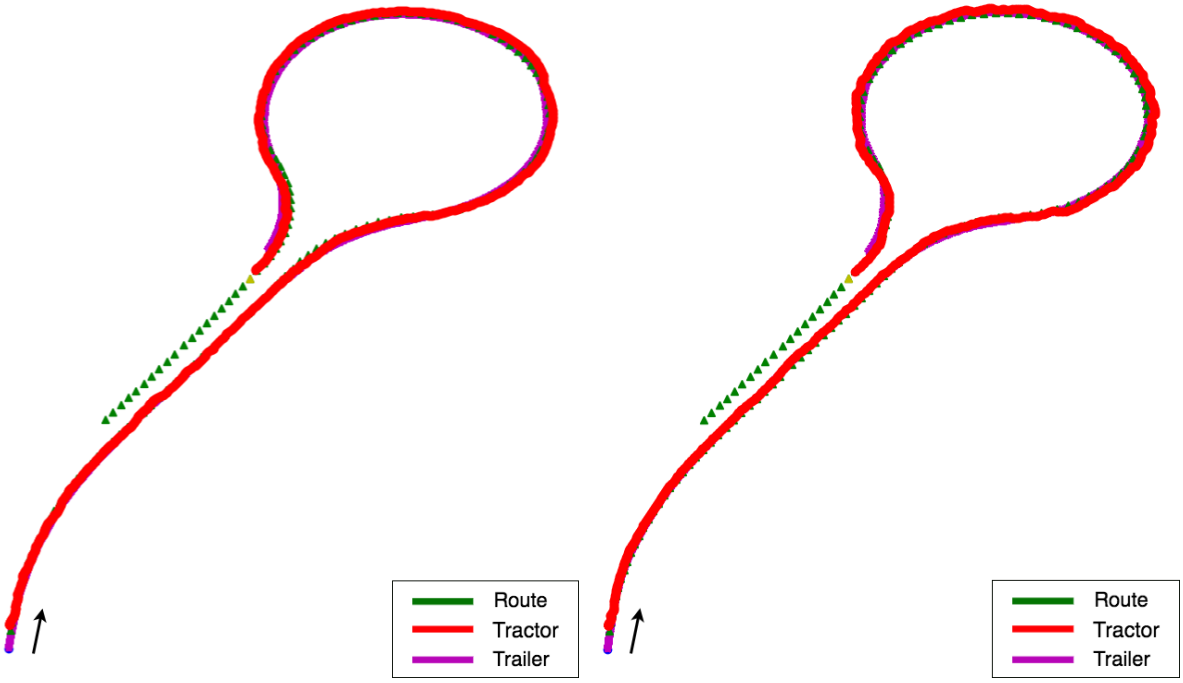


Figure 5.31 Forward velocity of the agent, obtained from two different models, of a route from the 40m testing roundabout.

As one can observe, the red plot, which is obtained from the model which includes the velocity smoothing reward function, has significantly smaller changes in velocity when compared to the blue plot, which does not make use of the velocity smoothing reward function. The velocity in the red plot rises to around $15km/h$ after which it steadily slows down to $12km/h$ despite some variations of around $3km/h$. When compared to the velocity graph in blue, the red plot is significantly more consistent and stable. Significant variations in velocity are observed in the blue plot where at one point the velocity drops from $9km/h$ to $3km/h$ and raises up to $10km/h$ in a short amount of timesteps, around 100 timesteps which represent 10 seconds. The velocity remains highly unstable throughout the whole route, dropping to $6km/h$ at the end. Given that the two plots represent the same route, one can observe the difference in the amount of timesteps taken by each model to complete the route. This is due to the higher velocity maintained by the 'variable.velocity.smoothing' model which managed to finish the route in around 400 less timesteps.

Observing the agent navigating the vehicle through the intersection using the 'variable.velocity.no.smoothing' model, one can observe a significant variance in velocities. The tractor-trailer vehicle almost comes to a complete stop before accelerating once again and continuing the route, which is highly undesirable behaviour. When using the 'variable.velocity.smoothing' model, a smoother driving behaviour is observed with only minor changes in velocity.



(a) 'variable.velocity.no.smoothing' model (b) 'variable.velocity.smoothing' model

Figure 5.32 Route followed by each mode from 40m testing roundabout.

Figure 5.32 visualises the routes followed by the 'variable.velocity.no.smoothing' and 'variable.velocity.smoothing' models. As one can observe, both models follow the path without any major deviation from the centre of the lane, with the red and magenta coloured paths, representing the tractor unit's and trailer unit's path respectively, being overlayed on top of the green waypoints. Both models are able to longitudinally follow the path optimally, despite having high and varying velocities.

While still maintaining optimal lateral control, further improvements can be made to the longitudinal control, obtaining more human-like behaviour. Ideal behaviour in a roundabout with no traffic would be to maintain a constant velocity, suitable for the curvature of the roundabout, throughout the whole roundabout, speeding up when the road straightens out. Further smoothing of the velocity, with the aim of reaching a constant velocity, can be obtained by increasing the weight vector of the smoothing reward function. Despite this, one has to ensure that increasing the negative reward due to the difference in velocities does not encourage the agent to settle for a lower velocity to minimise this negative reward.

5.4.2 Comparing the constant velocity and variable velocity action spaces

Comparing the constant velocity action space to the variable velocity action space allows one to evaluate the difference in behaviours these two models exhibit, where the latter is able to control the longitudinal behaviour of the vehicle as well as the lateral behaviour.

The constant velocity action space model used in this comparison is the 'Angles and Radii' model described in Section 5.2.2, which uses both the angles to the waypoints and the radius of the path alongside a weight vector of $\frac{1}{400}$ for the reward shaping function which determines the reward based to the distance to the centre of the lane. This model is referred to as 'constant.velocity'. The variable velocity action space model used in this comparison is the 'variable.velocity.smoothing' described in the previous section, due to its superior behaviour over the 'variable.velocity.no.smoothing' model. This model is from here onwards referred to as 'variable.velocity'. Both models make use of the PPO algorithm.

In order to correctly evaluate and compare the two models, the evaluation will be divided into the following subsections.

1. A quantitative evaluation including average training return graphs alongside evaluation metrics obtained from the models.
2. A qualitative evaluation assessing the unsuccessfully completed routes.
3. A qualitative evaluation assessing the successfully completed routes.
4. Summary of the findings

Quantitative Evaluation

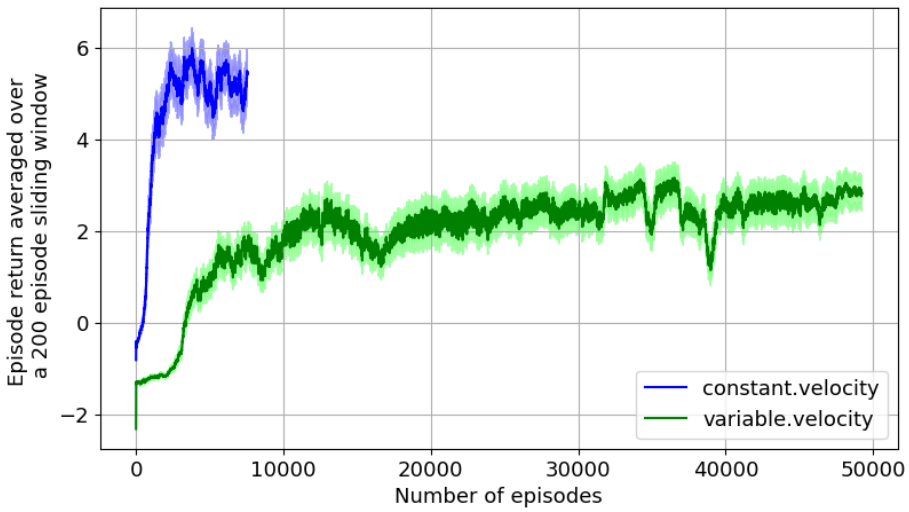


Figure 5.33 Average training return for the 'constant.velocity' and 'variable.velocity' models. The shaded area represents the 95% confidence interval.

Figure 5.33 illustrates the mean episode return averaged over a 200 episode sliding window for the 'constant.velocity' and 'variable.velocity' models. The 'constant.velocity'

Table 5.10 Training times for 'constant.velocity' and 'variable.velocity' models.

Models	Training Time (hours)
constant.velocity	4.2
variable.velocity	35.5

model is observed to initially increase its average return at a significantly faster rate, settling to around an average return of 5 after 8000 episodes. On the other hand, the 'variable.velocity' model increases its return at a lower rate, reaching an average return of 3 after almost 50000 episodes. The lower rate of return increase is due to the increased number of actions available to the model, which implies that more training is required to develop a suitable policy which chooses the correct action depending on the state.

Due to the additional negative reward included in the 'variable.velocity' model to smooth out the velocity changes, the lower average return obtained does not necessarily imply a lower success rate. Furthermore, one can observe the difference in training episodes required to reach a suitable model. The 'constant.velocity' model which only has a vector of 9 actions to choose from, reaches suitable convergence at around 8000 episodes. On the other hand, the 29 actions available to the 'variable.velocity' model, significantly increases the computational complexity required for the model to suitably converge at around 50000 episodes. As shown in Table 5.10, a difference in training time of 31 hours further illustrates the increased computation complexity required.

Table 5.11 Evaluation metrics on training roundabouts scenarios for the 'constant.velocity' and 'variable.velocity' models.

Training roundabout scenarios						
Model	Success Rate	Tractor Collision Rate	Trailer Collision Rate	d_c^t	d_c^{tt}	Timesteps
'constant.velocity'	0.81	0.05	0.14	0.61	0.43	625.7
'variable.velocity'	0.73	0.04	0.23	0.45	0.31	456.2

From the values in Table 5.11 for d_c^t and d_c^{tt}, a difference of 0.16m is observed for d_c^t and a difference of 0.12m is observed for d_c^{tt} when comparing the 'constant.velocity' and 'variable.velocity' models. As mentioned, these distances are computed only using values from the successfully completed routes are are averaged over all the routes. In order to obtain a clearer picture of how the distances to the centre of the lane varied for each route, similar statistical metrics to those utilised in Section 5.3 will be employed.

The average difference in distances d_c^t and d_c^{tt} for all routes which were at least successfully completed once, either by the 'constant.velocity' model or by the 'variable.velocity' model are, $\overline{\Delta d_c^t} = 0.13$ and $\overline{\Delta d_c^{tt}} = 0.1$. Since the difference was taken between the 'constant.velocity' model to the 'variable.velocity' model, positive values indicate average higher distances for the 'constant.velocity' model, while negative values indicate average higher distances for the 'variable.velocity' model.

An average Δd_c^t of 0.13m therefore indicates that on average, the tractor unit was 0.13m further away from the centre of the lane in the 'constant.velocity' model when compared to the 'variable.velocity' model. As discussed in Section 5.3, a difference in distance of 0.33m can be considered a significant difference when taking into consideration the whole context. Being less than 0.33m, the difference of 0.13m is considered insignificant. Similarly, an average difference of 0.1m when considering the trailer unit can be considered insignificant.

Using a cutoff point of 0.33m, further investigation shows that all the routes are considered to be insignificantly different in terms of lateral control. Some routes are observed to have almost identical distances to the centre of the lane while in others, the 'variable.velocity' model navigated the vehicle closer to the centre of the lane. Despite this, a maximum value of 0.32m indicates that for a specific route, the 'variable.velocity' model managed to navigate the tractor 0.32m closer to the centre of the lane with the trailer also being 0.32m closer to the centre of the lane.

Similar to the distance to the centre of the lane, the average of timesteps was computed only using the routes which have at least one successful run out of the 5 runs, in both models. This is to ensure that the average timestep value is computed only for routes which were successfully completed by both models and therefore unsuccessful routes which may have taken less timesteps, do not skew the results.

As one can observe in Table 5.11 the 'variable.velocity' model completes successful episodes 31% faster, with a difference of 169.5 timesteps or 16.95 seconds. As discussed above, this significant increase in velocity comes at the cost of a slightly lower success rate. Using such a simple reward function allows the agent to find the optimal velocity for each roundabout, depending on its physical properties and the characteristics of the vehicle being controlled.

Table 5.12 Evaluation metrics on testing roundabouts scenarios for the 'constant.velocity' and 'variable.velocity' models

	Testing roundabout scenarios					
Model	**Success Rate**	**Tractor Collision Rate**	**Trailer Collision Rate**	d_c^t	d_c^{tt}	**Timesteps**
'constant.velocity'	0.77	0.02	0.22	0.58	0.40	594.1
'variable.velocity'	0.68	0.05	0.28	0.45	0.34	404.9

Table 5.12 illustrates the evaluation metrics obtained from the testing roundabout scenarios for both the 'constant.velocity' and 'variable.velocity' models. Only 0.05 difference in success rate is observed when comparing the training and testing success rates of the 'variable.velocity' model. This indicates that the model has learnt to generalise the roundabout scenario and has not overfit to the training data. A similar decrease in success rate was observed for the 'variable.velocity' model.

As performed for the training roundabout scenarios, the difference in distances to the centre of the lane for each route are be evaluated. This obtains a clearer picture of how the different action spaces laterally controlled the vehicle.

The average difference in the distance of the tractor unit to the centre of the lane per route, between the 'constant.velocity' and 'variable.velocity' models, $\overline{\Delta d_c^t}$, is 0.13m. Taking a cutoff point at 0.33m, no route was observed to significantly deviate in terms of its lateral movement between the 'constant.velocity' and 'variable.velocity' models.

Despite this, one route obtained a difference d_c^t of 0.30m alongside a d_c^t of 0.00m. This means that the 'variable.velocity' model managed to navigate the tractor unit 0.30m closer the centre of the lane while maintaining the trailer the same distance away from the centre of the lane as the 'constant.velocity' model.

Similar to the training scenarios, the average timesteps only included routes which were at least completed once out of the 5 runs, in both models. A 38% increase in the number of timesteps per episode for the constant velocity model was noted. The 189 timestep difference, 18.9 seconds, is due to the higher velocity the 'variable.velocity' model navigates the tractor-trailer vehicle with.

Qualitative Evaluation - Unsuccessful Episodes

Table 5.11 illustrates the evaluation metrics obtained by both models on the training roundabout scenarios. A difference of 0.08 in success rate is observed between the two models. Further investigation of the training results shows that the 'variable.velocity' model failed to successfully complete the same routes failed by the 'constant.velocity' model, while also failing to complete some additional routes. Apart from some intermittent failures, all 5 episodes of 6 routes were unsuccessfully completed. These failed routes are from the 16m diameter roundabout, where in five of the routes, the tractor fails to sufficiently deviate away from the centre of the lane while navigating through a tight radius section, resulting in the tractor unit colliding with the inside kerb. The collision occurs in a similar situation to that shown in Figure 5.13, where the trailer is closer to the inside kerb. The other route which the 'variable.velocity' model failed to successfully complete is due to the tractor vehicle colliding head on with the splitting lane. This is illustrated in Figure 5.34. It should be noted that the splitting lane was purposely designed to increase the complexity of the path that needs to be taken by the tractor-trailer vehicle since following the centre of the lane while using this exit road leads to the trailer colliding the with splitting lane at the position marked by the red circle in Figure 5.34.

Visually analysing the 'variable.velocity' agent navigating through this route, one can observe that the tractor-trailer vehicle performs identical lateral movement when compared to the 'constant.velocity' agent. By analysing the velocity of the tractor-trailer

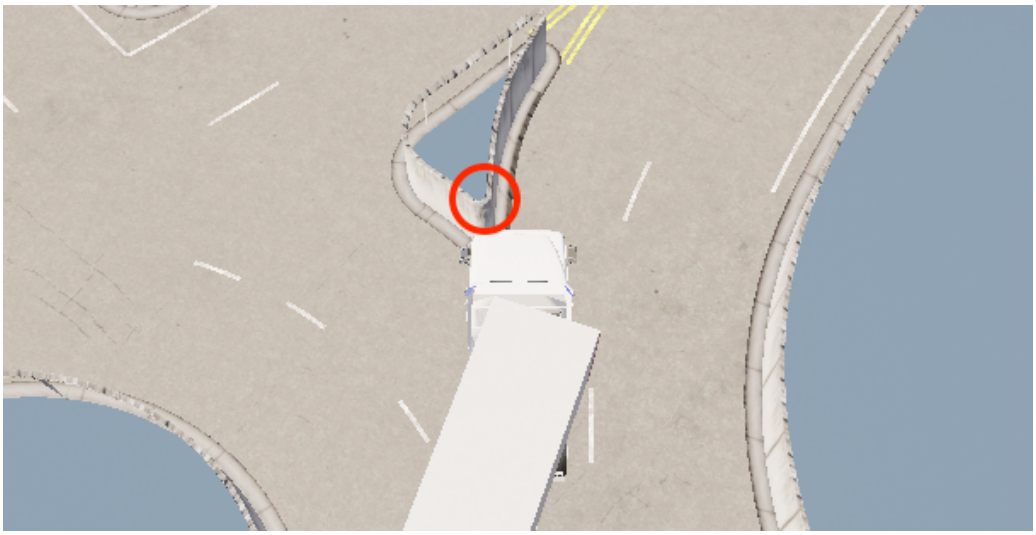


Figure 5.34 Tractor unit colliding headon with the splitting lane in the 20m roundabout.

vehicle utilised by both models for this route, one can give an explanation for the reason why the 'variable.velocity' model fails to successfully complete this route. Shown in Figure 5.35, one can observe that the 'variable.velocity' model maintains a higher velocity throughout the route of around $13km/h$ while the 'constant.velocity' model maintains a constant velocity of $8km/h$. At the final few timesteps of the 'variable.velocity' model, the velocity is observed to decrease. This implies that the agent notices the obstacle ahead and tried to slow the vehicle down to manoeuvre around the splitting lane. The distance sensors the agent uses to detect such obstacles have a range of 7m. Increasing such range could enable the agent to have a bigger picture of the surrounding environment. Furthermore, the relatively small footprint of the splitting lane decreases its detectability by the distance sensors.

From Figure 5.35, one can also notice the difference in time taken required to complete the episode where the 'variable.velocity' model, with its higher velocity, manages to almost complete the route in around 150 less timesteps when compared to the 'constant.velocity' model.

Evaluating the completed testing episodes shows that the 'variable.velocity' model manages to successfully complete a route which the 'constant.velocity' model always failed to complete. Apart from this, all routes failed by the 'constant.velocity' model were also unsuccessfully completed by the 'variable.velocity' model, with the 'variable.velocity' model intermittently failing other episodes. All testing routes on the 40m roundabout were successfully completed by the 'variable.velocity' model while the 20m roundabout proved to be more challenging.

The intermittent failing episodes are ones which out of the 5 runs, successfully reached the goal once. 4 routes of this kind were identified were, 2 only successfully completed the route once while the other 2 only successfully completed the route twice.

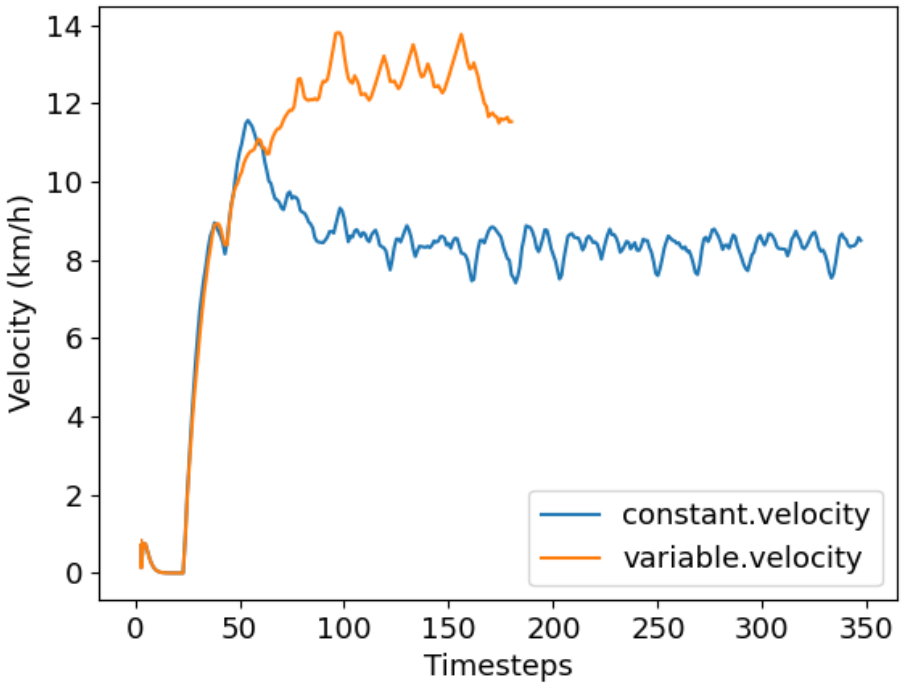


Figure 5.35 Velocity graphs for the route discussed in Figure 5.34 obtained the by 'constant.velocity' and 'variable.velocity' models.

Visually evaluating these failures reveals that most are due to the tractor-trailer vehicle not sufficiently deviating away from the centre of the lane while navigating through $270°$ or $360°$ of the roundabout. This leads to the trailer unit colliding with the inside kerb of the roundabout. The rest of the intermittent failures are due to a head on collision between the tractor unit and the splitting lane, similar to scenario showcased in Figure 5.34.

The failure point of the episode which the 'constant.velocity' model fails to complete while the 'variable.velocity' model successfully completes, is displayed in Figure 5.36. When comparing the control of both models, the 'variable.velocity' model deviates the truck further away from the centre of the lane in order to allow the trailer to clear the kerb.

Qualitative Evaluation - Successful Episodes

In the following part of this section, a trajectory analysis of the routes successfully navigated by the 'constant.velocity' and the 'variable.velocity' model will be performed.

Figure 5.37 illustrates the route in which the 'variable.velocity' model managed to navigate both the tractor unit and trailer unit 0.32m closer to the centre of the lane, when compared to the 'constant.velocity' model. In both inferences, the first section of the road was followed closely to the centre of the lane. This is shown by the red and

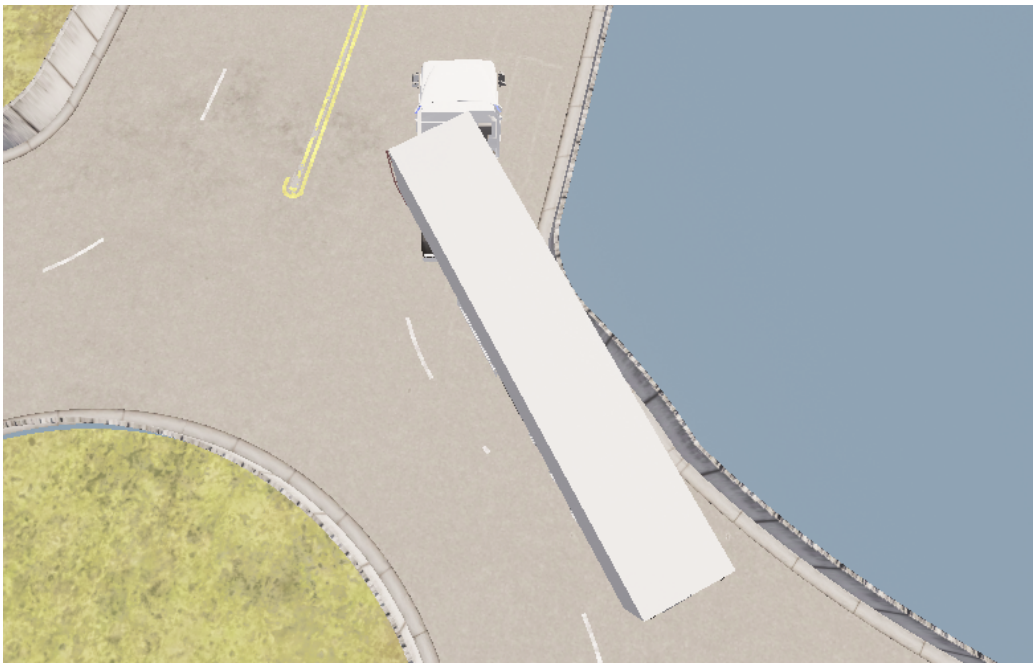
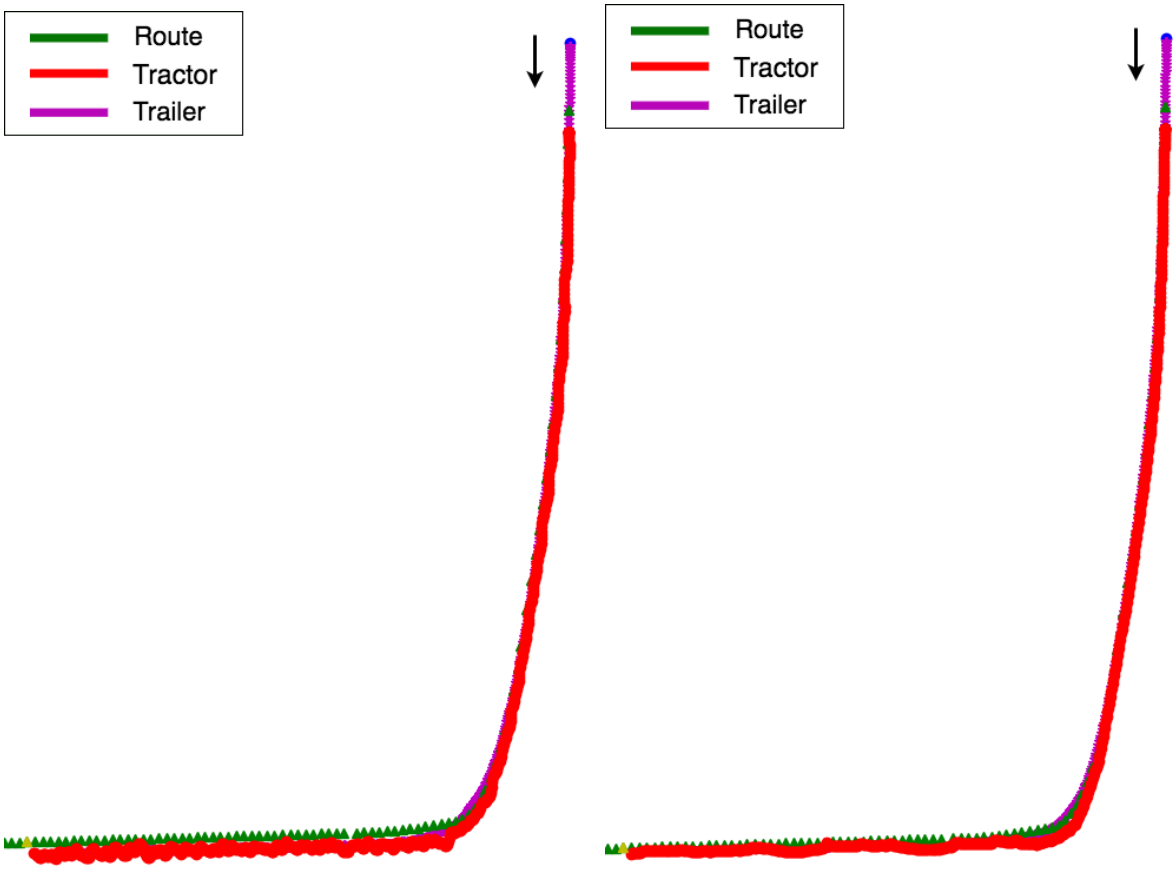


Figure 5.36 Failure point for ‘constant.velocity’ model on 20m testing roundabout route.



(a) ‘constant.velocity’ model

(b) ‘variable.velocity’ model

Figure 5.37 Visual of an identical route from the 32m training roundabout, navigated by different agents generated from different models.

magenta coloured paths, representing the tractor's and trailer's path, overlapping the green waypoints. After exiting the roundabout in the first exit, the following straight section of the road was navigated slightly differently by both models. In Figure 5.37a, the 'constant.velocity' model is observed to maintain the centre of the tractor and trailer units away from the centre of the lane, marked by the green waypoints. Furthermore, the unevenness of the red path indicates that the tractor unit was constantly changing its steering angle. A significantly smoother path, which is closer to the centre of the lane, is generated by the 'variable.velocity' model, as can be observed in Figure 5.37b. Such a smoother path may be attributed to the fact that the 'variable.velocity' model had a significantly higher number of training iterations and therefore, better convergence may have been obtained by this model.

Figure 5.38 compares the tractor unit's velocity throughout the route evaluated in Figure 5.37, for both the 'constant.velocity' and 'variable.velocity' models. Despite some variance in the velocity, the 'variable.velocity' model, maintains a higher overall velocity, completing the route in 200 less timesteps.

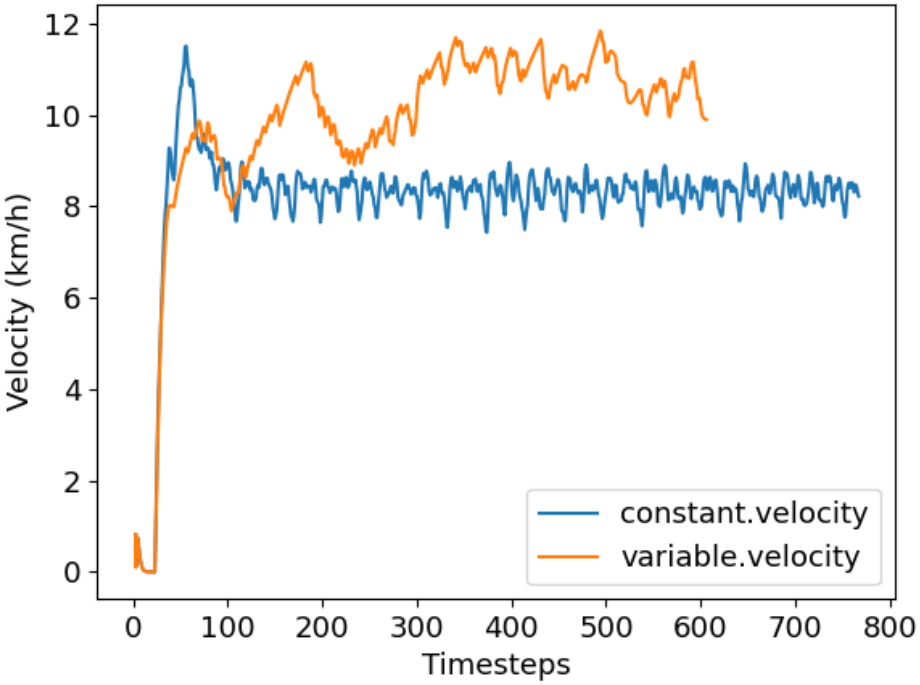


Figure 5.38 Velocity graph for the routes illustrated in Figure 5.37.

Figure 5.39 illustrates the navigation of a route from the 20m testing roundabout by the 'constant.velocity' and 'variable.velocity' model. For this route, the distance metrics Δd_c^t and Δd_c^{tt} were calculated at 0.06m and 0.09m respectively. Considering such a small difference in distances to the centre of the lane, the two routes can be said to be identical in terms of lateral navigation. This is confirmed by observing each route

separately as shown in Figures 5.39a and 5.39b. Both agents deviate slightly from the centre of the lane during the second half of the roundabout, with the remaining route being followed closely to the centre of the lane.

(a) 'constant.velocity' model (b) 'variable.velocity' model

Figure 5.39 Visual of an identical route from the 20m testing roundabout, navigated by different agents generated from different models.

The tractor unit's velocities maintained by the 'constant.velocity' and 'variable.velocity' models in the routes shown in Figure 5.39 are plotted in Figure 5.40. Once again, a significantly higher velocity is obtained by the 'variable.velocity' model where apart from successfully completing the episode 150 timesteps faster, the agent still maintains suitable lateral control of the vehicle as observed in Figure 5.39b.

It should be noted that, when using the variably.velocity model, none of the training or testing episode runs, failed to complete the episode due to timeout. This entails that the agent never remained stationary for more than 100 timesteps, despite the reward function not encouraging the agent to do so.

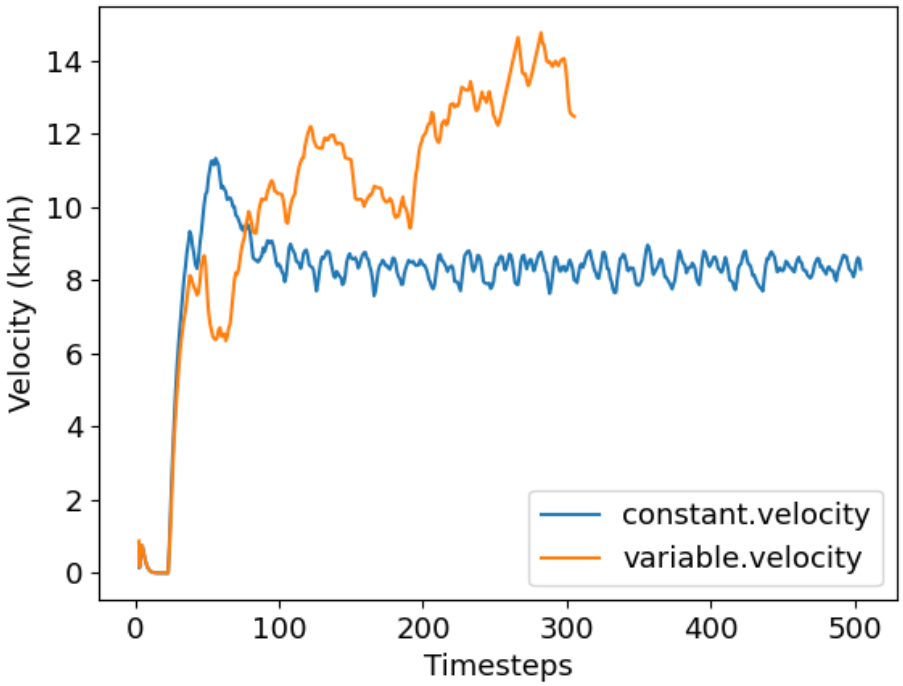


Figure 5.40 Velocity graph for the routes illustrated in Figure 5.39.

5.4.3 Comparison of results with current literature

Despite that the work of Diestra and Skouras [14], Zhang et al. [57], and Ren et al. [59], does not include the longitudinal control of the vehicle, comparison of the evaluation metrics can still be performed.

The 'constant.velocity' model, which is the PPO model mentioned in Section 5.3, is known to obtain higher training success rates and lower tractor and trailer deviation values when compared to previous research. The 'variable.velocity' model obtains a training success rate of 73% which is comparable to the 75% training success rate obtained by Ren et al. [59].

As can be observed in Figure 5.39b the trajectory taken by the tractor-trailer vehicle controlled by the 'variable.velocity' model is similar to that obtained by Diestra and Skouras [14] in Figure 5.18. Furthermore, as illustrated in Table 5.11, since the values of d_c^t and d_c^{tt} for the 'variable.velocity' model are smaller than those obtained by the 'constant.velocity' model, the 'variable.velocity' model can be noted to obtain smaller deviation values than current research.

Summary of Objective 4 Evaluation

This section compared two models, one made use of a PID controller to maintain a constant velocity, named 'constant.velocity' model, while the other was composed of

20 more actions which included varying acceleration values, named 'variable.velocity' model. Due to the larger action space, more computational complexity is required to obtain a suitable model. This is illustrated in the 31-hour difference when comparing the training times of both models. Only having a reward to obtain a smoother velocity throughout the route, with no capping on the maximum velocity, the 'variable.velocity' model varies the velocity with which the agent traverses through the roundabout intersection based on the roundabout. Navigating the vehicle using a velocity of, on average, $13km/h$, which is higher than the $8km/h$, the 'variable.velocity' model managed to complete episodes significantly faster. Despite the higher velocity, the difference in lateral control of both models can be considered insignificant for most routes, with only a handful of routes having a difference in distances to the centre of the lane greater than 0.2m.

Despite the difference in distances to the centre of the lane being considered insignificant, a lower success rate is obtained by the 'variable.velocity' model. The difference in distances indicate that the 'constant.velocity' model was further away from the centre of the lane. This further implies that the 'constant.velocity' model deviated away from the centre of the lane just enough since the small, considered insignificant, distance closer to the centre of the lane, obtained by the 'variable.velocity' model, was just enough for the trailer to collide in some of the episodes. Other collisions included the tractor unit colliding head on with the splitting lane due to the high velocity not allowing enough distance to steer away from it. The higher velocity can also be linked to the increase in trailer collisions, where the higher velocity does not allow the agent enough timesteps to make the appropriate steering changes. The higher velocity was maintained to a suitable level such that no episode failed due to the vehicle rolling over.

In our training and testing environments, only the success rate of smaller diameter roundabouts were affected when using the 'variable.velocity' model. Such roundabouts may have required slower velocities to be successfully navigated through, with 'variable.velocity' model failing to lower its velocity.

When comparing to previous literature, both the 'constant.velocity' and 'variable.velocity' models achieve comparable or superior training success rates while having lower lateral deviation values.

While the constant velocity action space focuses on taking advantage of the RL algorithm for longitudinal control, the variable velocity action space allows the control of both the lateral and longitudinal movements. Adding another dimension of control significantly increases the complexity of the problem but enables the two central controls of a vehicle to be controlled by one unit. This enables better control of the vehicle in cases where both the lateral and longitudinal movements need to work in synchrony. Despite not being a continuous action space, the 29 actions used in the 'variable.velocity' model gives an insight into the possibilities and limitations of using RL to control both the lateral and longitudinal movements of the tractor-trailer vehicle.

6 Conclusion

6.1 Revisiting the Aim and Objectives

The main aim of this work was to develop a reinforcement learning model to navigate a tractor-trailer vehicle through a roundabout intersection using lateral and longitudinal control. Due to the limited research related to tractor-trailer vehicles, no tractor and trailer vehicle models were publicly available in a high-fidelity simulator. Therefore, the first objective of this work was to create such models in the autonomous driving simulator, CARLA. Improving on the work of Engles[1] suitable visual and physical models for a tractor-trailer vehicle were developed. Furthermore, five different roundabout scenarios were developed. Each roundabout has varying physical properties, inspired by common roundabouts found in real-life road networks, testing the developed models' limits.

The second objective of this work consisted of developing a reinforcement learning framework to be able to navigate a tractor-trailer vehicle through a roundabout intersection using lateral and longitudinal movements. A vector of 69 continuous elements, including various information about the tractor-trailer vehicle and the route, was found to provide informative input observations. Furthermore, a reward function that included a reward shaping component was developed. Initially, a constant velocity action space was utilised to control the longitudinal movement of the vehicle using a PI controller, while the RL algorithm controlled the lateral movement.

In order to observe the difference in behaviour between different RL algorithms, PPO, DDDQN, and SAC models were evaluated. Such RL algorithms were chosen since they have significant differences and are considered state-of-the-art. While PPO uses an on-policy, policy gradient based approach, the DDDQN model is an off-policy, q-function based algorithm. The SAC algorithm is a hybrid between the two since it utilises the actor-critic approach, where both policy gradient and q-function approaches are used. When comparing these 3 algorithms, the DDDQN and SAC models were observed to significantly deviate away from the centre of the lane when deviation was not required. Further investigation concluded that these algorithms give more importance to input observations that give information about distant waypoints. This contrasts with the behaviour of the PPO model. Despite having a greater overall distance to the centre of the lane, the DDDQN and SAC models maintain the same distance away from the centre of the lane as the PPO model while the vehicle is circulating around the roundabout. Furthermore, on smaller diameter roundabouts, the PPO algorithm is observed to produce haphazard lateral movement. Despite the inconsistent steering behaviour, for

[1] www.github.com/frankeng/CarlaSemiTruckTrailer

our quasi-end-to-end autonomous driving problem, the on-policy, policy gradient PPO algorithm navigated the vehicle closer to human-like behaviour, producing the most optimal behaviour between the three RL algorithms.

The PPO algorithm trained using this framework obtained a training success rate of 0.81 and a testing success rate of 0.77. Such a model managed to successfully complete routes, which required a significant deviation away from the centre of the lane while also maintaining the vehicle close to the centre of the lane, mimicking human-like behaviour. Our PPO model obtained a superior success rate when compared to the work of Ren et al. [59], which obtained a training success rate of 0.75. Furthermore, when comparing to the work of Zhang et al. [57], our model obtained smaller lateral deviation values both for the tractor and trailer units.

We then explored the introduction of a variable velocity action space which enables the RL algorithm to control the longitudinal movement of the vehicle. Despite adding a reward shaping function to smooth out the velocity, since no limitation on the maximum velocity that can be reached was imposed, the 'variable.velocity' model manages to complete episodes 38% faster than the 'constant.velocity' model. This higher velocity also resulted in a greater number of collisions where the tractor unit failed to avoid colliding with a small splitting lane. In spite of this, the 'variable.velocity' model managed to laterally control the tractor-trailer vehicle very similar to the 'constant.velocity' and in most cases, obtaining smoother lateral movements.

All four objectives and the initial aim of this work were successfully achieved. An RL based model for controlling the longitudinal and lateral movement of the vehicle through a roundabout intersection was developed. The PPO model managed to maintain the tractor-trailer vehicle close to the centre of the lane while also being able to deviate away from the centre of the lane when encountering a low curvature path.

6.2 Future Work

The developed models are observed to experience a significant amount of variance. Further investigation is required to understand where each policy fails and decrease such variance.

Autonomous navigation algorithms are designed to be deployed in real-world scenarios. Such real-world scenarios significantly vary in complexity and therefore these algorithms should be able to handle varying scenarios.

Further developing the roundabout dataset to include more low-radius right-hand turns, increases the model's generalisability to such manoeuvres enabling better performance. Real-life roundabout intersections necessitate the interaction between road users in order to safely navigate the intersection while causing the least disruption to

the circulating traffic. The introduction of traffic would test the agent's cooperation with other vehicles which may also behave differently to each other. Furthermore, since the tractor-trailer vehicle may occupy more than one lane, additional awareness of the surrounding vehicles is required.

While the introduction of traffic affects which lane the vehicle can occupy at a specific point in time, the addition of obstacles throughout the roundabout can demonstrate the advantages of using an RL algorithm which can adapt to unseen and changing environments. The roundabout dataset can also be further developed by introducing roundabouts with a higher number of lanes alongside more specific roundabouts such as double mini roundabouts. Such scenarios increase the confidence of the generalisability of the developed models, by training and testing them on a variety of different roundabouts. Such roundabouts may vary in their maximum velocities, introducing new failure cases for tractor-trailer vehicles such as jackknifing and rollover.

When dealing with roundabout intersections and by using a high-fidelity simulator, a major advantage of allowing the RL to control the longitudinal behaviour of the vehicle is that the model can learn the optimal velocity for different roundabouts based on their curvature while also taking into consideration the physical constraints of the tractor-trailer vehicle. Therefore, future research can work towards ensuring that the reward function, specifically the reward shaping function, does not limit the ability of the agent to find the optimal longitudinal control.

In this research, only tractor-trailer vehicles are considered. Different articulated vehicles including multiple trailer units connected to a single tractor can be investigated. Furthermore, using the trailer's distance to the centre of the lane may be tested as a reward function component. Furthermore, aiming to ensure robustness, future research may delve into adding a safety guarantee layer on the output of the RL algorithm to ensure that the chosen action is suitable and will not cause a collision.

On a more general level, different weather conditions, such as rain, can be simulated to evaluate the behaviour of the model when different traction forces are exerted. Furthermore, the developed roundabout scenarios are approached by flat roads. Varying the inclination of the entry-exit paths implies that the vehicle experiences different acceleration values which in turn affects its navigation. Such acceleration values value are also heavily affected by the weight of the trailer. The distance sensor data utilised in this research was 100% accurate but the same cannot be said for real-world data. The introduction of noise in the collected sensor data increases the model's robustness, required when it is applied in real-life environments. Developing our work in the high-fidelity simulator CARLA, enables such changes with minimal effort.

While aiming to obtain robust and reliable models, minimizing the state space dimensions allows for faster inference while also reducing the computational complexity and therefore environmental impact when training such models.

6.3 Final Remarks

The motivation for this work was to develop an RL environment to navigate a tractor-trailer vehicle through a roundabout intersection. Such a system aims to reduce carbon emissions by improving fuel efficiency, increase road safety by reducing the amount of road traffic accidents, and increase the efficiency of the logistics sector. Research relating to this field is relatively sparse and therefore the development of an RL environment which can be used for the autonomous navigation of tractor-trailer vehicles was crucial. Furthermore, this work also developed a tractor-trailer vehicle in the high-fidelity simulation software CARLA and a dataset of roundabout intersections. Such artefacts are publicly available on GitHub[2] and can be used to accelerate future research in this field, with the dataset serving as a benchmark, enabling comparison between models.

[2] https://github.com/DanielAtt2000/Tractor-Trailer-Vehicle-and-Roundabout-Dataset-Carla

References

[1] World Health Organisation, “Road traffic injuries,” Tech. Rep., Jun. 2022. [Online]. Available: https://www.who.int/news-room/fact-sheets/detail/road-traffic-injuries.

[2] National Centre for Statistics and Analysis, “2016 Fatal Motor Vehicle Crashes: Overview,” U.S Department of Transportation, Washington, USA, Tech. Rep., Oct. 2017. [Online]. Available: https://crashstats.nhtsa.dot.gov/Api/Public/Publication/812456.

[3] OECD/ITF (2015), “Road Safety Annual Report 2015,” OECD Publishing, Paris, Tech. Rep., Oct. 2015. [Online]. Available: http://dx.doi.org/10.1787/irtad-2015-en.

[4] Bureau of Transportation Statistics, “U.S. Vehicle-Miles,” Bureau of Transportation Statistics, USA, Tech. Rep., 2023. [Online]. Available: https://www.bts.gov/product/national-transportation-statistics.

[5] E. Kim, Y. Kim, and J. Park, “The Necessity of Introducing Autonomous Trucks in Logistics 4.0,” *Sustainability*, vol. 14, no. 7, p. 3978, Mar. 2022. DOI: 10.3390/su14073978.

[6] K. Bullis, *”How Vehicle Automation Will Cut Fuel Consumption,” MIT Technology Review*, 2011. [Online]. Available: https://www.technologyreview.com/2011/10/24/190538/how-vehicle-automation-will-cut-fuel-consumption/ (visited on 05/05/2023).

[7] J. Short and D. Murray, “Identifying Autonomous Vehicle Technology Impacts on the Trucking Industry,” American Transportation Research Institute, USA, Tech. Rep., 2016.

[8] M. K. Sadeghi, “Definitions of Performance Based Characteristics for Long Heavy Vehicle Combinations,” Chalmers University of Technology, Gothenburg, Sweden, Tech. Rep., Jun. 2013. [Online]. Available: https://research.chalmers.se/en/publication/176464.

[9] K. Jo, J. Kim, D. Kim, C. Jang, and M. Sunwoo, “Development of Autonomous Car—Part I: Distributed System Architecture and Development Process,” *IEEE Transactions on Industrial Electronics*, vol. 61, no. 12, pp. 7131–7140, Dec. 2014. DOI: 10.1109/TIE.2014.2321342.

[10] J. Chen, S. E. Li, and M. Tomizuka, "Interpretable End-to-end Urban Autonomous Driving with Latent Deep Reinforcement Learning," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 6, pp. 5068–5078, Jun. 2022. DOI: 10.1109/TITS.2020.3046646.

[11] Z. Liu, A. Amini, S. Zhu, S. Karaman, S. Han, and D. L. Rus, "Efficient and Robust LiDAR-Based End-to-End Navigation," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*, 2021, pp. 13 247–13 254. DOI: 10.1109/ICRA48506.2021.9561299.

[12] A. Capasso, G. Bacchiani, and D. Molinari, "Intelligent Roundabout Insertion using Deep Reinforcement Learning," in *Proceedings of the 12th International Conference on Agents and Artificial Intelligence - Volume 2: ICAART*, SciTePress, 2020, pp. 378–385. DOI: 10.5220/0008915003780385.

[13] F. Konstantinidis, U. Hofmann, M. Sackmann, J. Thielecke, O. De Candido, and W. Utschick, "Parameter Sharing Reinforcement Learning for Modeling Multi-Agent Driving Behavior in Roundabout Scenarios," in *2021 IEEE International Intelligent Transportation Systems Conference (ITSC)*, 2021, pp. 1974–1981. DOI: 10.1109/ITSC48978.2021.9565031.

[14] J. Diestra and S. Skouras, "Trajectory Planning for Automated Driving of Articulated Heavy Vehicles," M.S. thesis, Department of Electrical Engineering, Chalmers University of Technology, Gothenburg, Sweden, 2019.

[15] R. Oliveira, O. Ljungqvist, P. F. Lima, and B. Wahlberg, "A Geometric Approach to On-road Motion Planning for Long and Multi-Body Heavy-Duty Vehicles," in *2020 IEEE Intelligent Vehicles Symposium (IV)*, Oct. 2020, pp. 999–1006. DOI: 10.1109/IV47402.2020.9304767.

[16] I.-M. Chen and C.-Y. Chan, "Deep reinforcement learning based path tracking controller for autonomous vehicle," *Proceedings of the Institution of Mechanical Engineers, Part D: Journal of Automobile Engineering*, vol. 235, no. 2-3, pp. 541–551, 2021. DOI: 10.1177/0954407020954591.

[17] L. Jiang, Y. Wang, L. Wang, and J. Wu, "Path tracking control based on Deep reinforcement learning in Autonomous driving," in *2019 3rd Conference on Vehicle Control and Intelligence (CVCI)*, 2019, pp. 1–6. DOI: 10.1109/CVCI47823.2019.8951665.

[18] X. Lin, J. Zhang, J. Shang, Y. Wang, H. Yu, and X. Zhang, "Decision Making through Occluded Intersections for Autonomous Driving," in *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*, Oct. 2019, pp. 2449–2455. DOI: 10.1109/ITSC.2019.8917348.

[19] H. Liu, Z. Huang, J. Wu, and C. Lv, "Improved Deep Reinforcement Learning with Expert Demonstrations for Urban Autonomous Driving," in *2022 IEEE Intelligent Vehicles Symposium (IV)*, 2022, pp. 921–928. DOI: 10.1109/IV51971.2022.9827073.

[20] Y. Li, *Deep Reinforcement Learning*, arXiv:1810.06339, 2018.

[21] A. P. Capasso, G. Bacchiani, and A. Broggi, "From Simulation to Real World Maneuver Execution using Deep Reinforcement Learning," in *2020 IEEE Intelligent Vehicles Symposium (IV)*, 2020, pp. 1570–1575. DOI: 10.1109/IV47402.2020.9304593.

[22] M. Zhao, D. Kathner, M. Jipp, D. Soffker, and K. Lemmer, "Modeling driver behavior at roundabouts: Results from a field study," in *2017 IEEE Intelligent Vehicles Symposium (IV)*, 2017, pp. 908–913.

[23] Y. Zhang, B. Gao, L. Guo, H. Guo, and H. Chen, "Adaptive Decision-Making for Automated Vehicles Under Roundabout Scenarios Using Optimization Embedded Reinforcement Learning," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 32, no. 12, pp. 5526–5538, 2021. DOI: 10.1109/TNNLS.2020.3042981.

[24] L. García Cuenca, E. Puertas, J. Fernandez Andrés, and N. Aliane, "Autonomous Driving in Roundabout Maneuvers Using Reinforcement Learning with Q-Learning," *Electronics*, vol. 8, no. 12, p. 1536, 2019. DOI: 10.3390/electronics8121536.

[25] M. A. Serin and T. Soodla, "Tactical Decision-Making for Heavy Vehicles using Deep Reinforcement Learning," M.S. thesis, Department of Electrical Engineering, Chalmers University of Technology, Gothenburg, Sweden, 2020.

[26] P. Wolf *et al.*, "Learning how to drive in a real world simulation with deep Q-Networks," in *2017 IEEE Intelligent Vehicles Symposium (IV)*, 2017, pp. 244–250. DOI: 10.1109/IVS.2017.7995727.

[27] Z. Wang, X. Liu, and Z. Wu, "Design of Unsignalized Roundabouts Driving Policy of Autonomous Vehicles Using Deep Reinforcement Learning," *World Electric Vehicle Journal*, vol. 14, no. 2, p. 52, 2023. DOI: 10.3390/wevj14020052.

[28] D. Wang *et al.*, "An Intelligent Self-driving Truck System For Highway Transportation," *Front Neurorobot*, vol. 16, 2022.

[29] 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)*, 2019, pp. 1–4. DOI: 10.1109/CCECE.2019.8861534.

[30] E. Olcay, X. Rui, and R. Wang, “Headland Turn Automation Concept for Tractor-Trailer System with Deep Reinforcement Learning,” in *2023 IEEE 19th International Conference on Automation Science and Engineering (CASE)*, 2023, pp. 1–7. DOI: 10.1109/CASE56687.2023.10260531.

[31] National Heavy Vehicle Regulator, “Performance Based Standards 2.0,” National Heavy Vehicle Regulator, Tech. Rep., Jun. 2022.

[32] A. G. Sutton Richard S. and Barto, *Reinforcement Learning: An Introduction*, Second. The MIT Press, 2018.

[33] P. Brémaud, *Markov Chains: Gibbs Fields, Monte Carlo Simulation and Queues* (Texts in Applied Mathematics). Springer, 2020, vol. 31.

[34] X. Fang, S. Misra, G. Xue, and D. Yang, “Smart Grid — The New and Improved Power Grid: A Survey,” *IEEE Communications Surveys & Tutorials*, vol. 14, no. 4, pp. 944–980, 2012. DOI: 10.1109/SURV.2011.101911.00087.

[35] P.-Y. Oudeyer, J. Gottlieb, and M. Lopes, “Chapter 11 - Intrinsic motivation, curiosity, and learning: Theory and applications in educational technologies,” in *Motivation*, ser. Progress in Brain Research, vol. 229, Elsevier, 2016, pp. 257–284.

[36] A. Rajkomar *et al.*, “Scalable and accurate deep learning with electronic health records,” *npj Digital Medicine*, vol. 1, no. 1, p. 18, 2018. DOI: 10.1038/s41746-018-0029-1.

[37] K. Eykholt *et al.*, “Robust Physical-World Attacks on Deep Learning Visual Classification,” in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2018, pp. 1625–1634. DOI: 10.1109/CVPR.2018.00175.

[38] V. Mnih *et al.*, “Asynchronous Methods for Deep Reinforcement Learning,” in *Proceedings of The 33rd International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, vol. 48, New York, USA: PMLR, 2016, pp. 1928–1937.

[39] J. Schulman, S. Levine, P. Moritz, M. I. Jordan, and P. Abbeel, “Trust Region Policy Optimization,” in *Proceedings of the 32nd International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, vol. 37, Lille, France: PMLR, 2015, pp. 1889–1897.

[40] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, *Proximal Policy Optimization Algorithms*, arXiv:1707.06347, 2017.

[41] V. Mnih *et al.*, “Human-level control through deep reinforcement learning,” *Nature*, vol. 518, no. 7540, pp. 529–533, 2015. DOI: 10.1038/nature14236.

[42] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, *Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor*, arXiv:1801.01290, 2018.

[43] V. Feinberg, A. Wan, I. Stoica, M. I. Jordan, J. E. Gonzalez, and S. Levine, *Model-Based Value Estimation for Efficient Model-Free Reinforcement Learning*, arXiv:1803.00101, 2018.

[44] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015. DOI: 10.1038/nature14539.

[45] E. Carvalho *et al.*, "Exploiting the use of recurrent neural networks for driver behavior profiling," in *2017 International Joint Conference on Neural Networks (IJCNN)*, 2017, pp. 3016–3021. DOI: 10.1109/IJCNN.2017.7966230.

[46] D. Silver *et al.*, "Mastering the game of Go with deep neural networks and tree search," *Nature*, vol. 529, no. 7587, pp. 484–489, 2016. DOI: 10.1038/nature16961.

[47] Y. Zhu *et al.*, "Target-driven visual navigation in indoor scenes using deep reinforcement learning," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, 2017, pp. 3357–3364. DOI: 10.1109/ICRA.2017.7989381.

[48] G. Tesauro *et al.*, "Managing Power Consumption and Performance of Computing Systems Using Reinforcement Learning," in *Proceedings of the 20th International Conference on Neural Information Processing Systems*, ser. NIPS'07, vol. 20, Curran Associates, Inc., 2007, pp. 1497–1504.

[49] H. Van Hasselt, A. Guez, and D. Silver, "Deep Reinforcement Learning with Double Q-Learning," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 30, no. 1, 2016. DOI: 10.1609/aaai.v30i1.10295.

[50] T. P. Lillicrap *et al.*, *Continuous control with deep reinforcement learning*, arXiv:1509.02971, 2019.

[51] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, "CARLA: An Open Urban Driving Simulator," in *Proceedings of the 1st Annual Conference on Robot Learning*, 2017, pp. 1–16.

[52] P. Cai, S. Wang, H. Wang, and M. Liu, *Carl-Lead: Lidar-based End-to-End Autonomous Driving with Contrastive Deep Reinforcement Learning*, arXiv:2109.08473, 2021.

[53] Z. Cao, S. Xu, H. Peng, D. Yang, and R. Zidek, "Confidence-Aware Reinforcement Learning for Self-Driving Cars," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 7, pp. 7419–7430, 2022. DOI: 10.1109/TITS.2021.3069497.

[54] J. P. Rastelli and M. S. Peñas, "Fuzzy logic steering control of autonomous vehicles inside roundabouts," *Applied Soft Computing*, vol. 35, pp. 662–669, 2015. DOI: 10.1016/j.asoc.2015.06.030.

[55] J. Chen, B. Yuan, and M. Tomizuka, "Model-free Deep Reinforcement Learning for Urban Autonomous Driving," in *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*, 2019, pp. 2765–2771. DOI: 10.1109/ITSC.2019.8917306.

[56] A. Widyotriatmo, P. I. Siregar, and Y. Y. Nazaruddin, "Line following control of an autonomous truck-trailer," in *2017 International Conference on Robotics, Biomimetics, and Intelligent Computational Systems (Robionetics)*, 2017, pp. 24–28. DOI: 10.1109/ROBIONETICS.2017.8203431.

[57] X. Zhang, J. Eck, and F. Lotz, "A Path Planning Approach for Tractor-Trailer System based on Semi-Supervised Learning," in *2022 IEEE 25th International Conference on Intelligent Transportation Systems (ITSC)*, Oct. 2022, pp. 3549–3555. DOI: 10.1109/ITSC55140.2022.9922552.

[58] P. A. Lopez *et al.*, "Microscopic Traffic Simulation using SUMO," in *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, 2018, pp. 2575–2582. DOI: 10.1109/ITSC.2018.8569938.

[59] Z. Ren *et al.*, "Double-DQN-Based Path-Tracking Control Algorithm for Orchard Traction Spraying Robot," *Agronomy*, vol. 12, no. 11, p. 2803, Nov. 2022. DOI: 10.3390/agronomy12112803.

[60] W. B. Knox, A. Allievi, H. Banzhaf, F. Schmitt, and P. Stone, "Reward (Mis)design for Autonomous Driving," *Artificial Intelligence*, vol. 316, 2023. DOI: https://doi.org/10.1016/j.artint.2022.103829.

[61] G. Rong *et al.*, "LGSVL Simulator: A High Fidelity Simulator for Autonomous Driving," in *2020 IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC)*, 2020, pp. 1–6. DOI: 10.1109/ITSC45102.2020.9294422.

[62] Council of European Union, *Council Directive 96/53/EC - Laying down for certain road vehicles circulating within the Community the maximum authorized dimensions in national and international traffic and the maximum authorized weights in international traffic*, 1996.

[63] MATLAB, *version 9.13.0.20 (R2022b)*. Natick, Massachusetts: The MathWorks Inc., 2022.

[64] E. Liang *et al.*, "RLlib: Abstractions for Distributed Reinforcement Learning," in *Proceedings of the 35th International Conference on Machine Learning*, vol. 80, PMLR, 2018, pp. 3059–3068.

[65] M. Towers *et al.*, *Gymnasium*, Mar. 2023. DOI: 10.5281/zenodo.8127026. [Online]. Available: https://zenodo.org/record/8127025 (visited on 07/08/2023).

[66] J. G. Ziegler and N. B. Nichols, “Optimum Settings for Automatic Controllers,” *Journal of Dynamic Systems, Measurement, and Control*, vol. 115, no. 2B, pp. 220–222, 1993. DOI: 10.1115/1.2899060.