Universiteit
Leiden
The Netherlands

Bachelor DS & AI

RL in a dynamic environment

Jan Hendrik van Veen

First supervisor:
Aske Plaat
Second supervisor:
Álvaro Serra-Gómez

BACHELOR THESIS

Leiden Institute of Advanced Computer Science (LIACS)
www.liacs.leidenuniv.nl

June 19, 2025

Abstract

Human error remains a leading cause of road accidents, underscoring the importance of developing autonomous vehicles capable of safe navigation. This study investigates the training of reinforcement learning (RL) agents for autonomous driving in the CARLA simulator, with a focus on collision avoidance. We implement and compare three approaches: Proximal Policy Optimisation (PPO), Soft Actor-Critic (SAC), and a hybrid PPO combined with Model Predictive Control (PPO+MPC). Each agent is trained using a reward function that emphasises collision avoidance. Performance is evaluated based on safety metrics, learning efficiency, and adaptability to new scenarios. Preliminary expectations suggest that while model-free methods such as PPO and SAC may excel in complex environments, the hybrid PPO+MPC approach could offer advantages in structured settings. However, after comparing the three approaches, the results were unexpected: both the PPO agent and the PPO+MPC hybrid underperformed, while SAC demonstrated the most promising results in terms of safety and adaptability.

Contents

1 Introduction

1.1 Topic

Self-driving cars have captured the imagination of both the public and researchers alike, combining sleek technology with the promise of safer, more efficient transportation. These vehicles are equipped with advanced sensors, cameras, and intelligent software that enable them to navigate complex environments autonomously. Among the many approaches to enable autonomy, Reinforcement Learning (RL) has emerged as a powerful technique for teaching cars to make decisions in real time. RL is a branch of machine learning in which an agent learns to make decisions by interacting with its environment, receiving feedback in the form of rewards or penalties. The agent's goal is to maximise cumulative rewards, effectively learning an optimal policy for decision making. In the real world, however, there is no such thing as an optimal policy, since the real world is dynamic and thus no state is the same. This makes RL very difficult to use in a real-world environment.

1.2 Problem statement

The objective of this study is to develop and compare various RL algorithms to enable a self-driving car to navigate from a predefined start point to a destination while avoiding collisions in a dynamic environment simulated using CARLA [Dosovitskiy et al., 2017]. The research seeks to provide information on the effectiveness of different RL approaches for autonomous driving tasks, with a particular focus on collision avoidance and decision making in dynamic conditions.
To achieve this, the study is guided by the following central research question.
Problem Statement:

- Can a RL agent be trained to navigate autonomously in CARLA based on collision avoidance?

This overarching inquiry is further explored through the following sub-questions:

RQ1. What RL algorithms are most effective for collision avoidance in dynamic environments?
Given the critical importance of collision avoidance in autonomous driving, this question aims to identify which RL algorithms are most capable of enabling safe navigation in environments with dynamic obstacles.
RQ2. How can different reward structures let the agent find the route to its destination?
In addition to avoiding collisions, an effective autonomous agent must successfully navigate to its destination. This question explores how various reward designs impact the agent's route-finding behaviour and goal-oriented performance.
RQ3. Does the use of MPC with an RL algorithm improve its performance? MPC is known for its effectiveness in handling short-term, reactive manoeuvres such as emergency braking. This question investigates whether combining MPC with an RL algorithm improves the agent's overall driving performance, particularly in scenarios requiring rapid decision-making.

1.3 Method

To investigate the research problem, I will perform an experiment using a simulated self-driving car within the CARLA simulator. The focus of the experiment will be on applying RL techniques

to train an autonomous driving agent capable of navigating complex environments. The agent will interact with the simulation to learn driving behaviours through trial and error, guided by a reward function designed to encourage safe driving. This experimental setup enables controlled, repeatable testing and provides a foundation for analysing the capabilities of RL in autonomous driving applications.

1.4 Overview

The next chapter provides a comprehensive review of the existing literature pertinent to this thesis, concentrating on key algorithms, prior experimental studies, and the simulation environments commonly employed in this domain. This chapter is structured into distinct sections covering algorithms, simulation environments, and Model Predictive Control (MPC), offering a foundation for the research conducted. Chapter three elaborates on the experimental methodology, detailing the implementation and training of multiple algorithms within the Carla simulation environment. This section emphasises the hands-on development process and the design choices made to compare algorithmic performance. Following this, the results chapter presents a detailed visualisation of the experimental outcomes, primarily through graphs, which illustrate significant performance variations as well as some unexpected behaviours. The thesis concludes with a reflective discussion that synthesises the findings, critically evaluates the research approach, and explores the limitations encountered. Additionally, it outlines possible avenues for improvement and suggests directions for future work, thereby situating the contribution within a broader research context.

2 Related work

This chapter reviews the existing body of research relevant to the scope of this thesis. It begins by examining key algorithms that have shaped the development of the methods under consideration. Next, the chapter explores the various simulation environments commonly used to evaluate these algorithms, with particular attention to their features and limitations. Finally, it discusses prior work related to MPC, highlighting its applications and relevance to the current study. Together, these sections provide a comprehensive background and context for the experimental work that follows.

2.1 Algorithms

A fundamental theoretical grounding for RL is laid out in the seminal textbook Reinforcement Learning: An Introduction by Richard S. Sutton and Andrew G. Barto [Sutton and Barto, 1998]. This work introduces the core concepts of RL, such as the Markov Decision Process (MDP) framework, value functions, policy iteration, and temporal-difference learning. The book provides rigorous mathematical foundations alongside conceptual clarity, making it a cornerstone in the field. However, the focus is predominantly on stationary environments with fixed dynamics. The classical RL frameworks described by Sutton and Barto assume a stable world model, which is often not reflective of many real-world applications where agents must adapt to changing conditions. This thesis builds on the principles introduced in this foundational work but extends them to scenarios characterised by dynamic or non-stationary environments, necessitating the development or application of more adaptive learning strategies.

More recent advances in deep RL are surveyed by Yuxi Li in Deep Reinforcement Learning: An Overview [Li, 2017]. The paper outlines key algorithmic classes—value-based, policy-based, and actor-critic methods—and highlights challenges such as stability, exploration, and sample efficiency. While this survey provides an overview of deep RL up to 2017, it largely centres on static environments.

In their study, Muzahid et al. [2021] compared the performance of Proximal Policy optimisation (PPO) and Soft Actor-Critic (SAC) algorithms in multi-agent scenarios for collision avoidance among autonomous vehicles. The research demonstrated that both algorithms achieved comparable success rates, with PPO showing faster convergence and SAC offering more stable performance in dynamic environments. While their focus was on multi-agent interactions in a 2D setting, the insights into each algorithm's strengths are pertinent to this thesis.

This research extends the application of PPO and SAC to a single-agent context within a complex 3D environment, where the agent must navigate and adapt to dynamic changes. The choice of these algorithms is informed by their demonstrated capabilities in handling dynamic scenarios, as evidenced by Muzahid et al. [2021]. By evaluating their performance in a more complex spatial setting, this study aims to assess their adaptability and efficiency in single-agent, high-dimensional tasks.

2.2 Environment

The paper "SAC-RL: Continuous Control of Wheeled Mobile Robot for Navigation in a Dynamic Environment" [Sharma, 2020] presents a RL approach utilising the SAC algorithm for autonomous

navigation of wheeled mobile robots in dynamic settings. The authors design a reward function that balances goal-reaching efficiency with obstacle avoidance, incorporating penalties for collisions and deviations from the desired path. To manage complex environments, the study employs sub-goal setting, where intermediate targets are defined approximately every 4 metres, simplifying the navigation task into manageable segments. This work directly addresses the challenges of dynamic environments by enabling the robot to adapt its trajectory in response to moving obstacles and changing conditions. The combination of SAC's entropy-based exploration and the structured reward function allows the robot to learn policies that are both safe and efficient in non-static settings. Building upon the insights from this study, the current research adopts a simplified approach to reward structuring and goal setting. Recognising that the primary objective for autonomous vehicles is to avoid collisions, the reward function is streamlined to heavily penalise collisions while being more permissive of off-path movements, such as driving off-road or in unintended directions, if it ensures safety. Additionally, instead of predefining sub-goals at fixed intervals, this work explores a dynamic goal-updating mechanism, where the goal position is incrementally moved as the agent progresses. This strategy aims to encourage the agent to plan over longer horizons and adapt more fluidly to the environment, potentially leading to more robust navigation behaviours in complex, dynamic 3D settings.

Given the inherent risks and logistical challenges of conducting real-world experiments in autonomous driving, this study employs CARLA (Car Learning to Act), an open-source simulator specifically designed for autonomous driving research [Dosovitskiy et al., 2017]. CARLA provides a high-fidelity, urban driving environment that supports the development, training, and validation of autonomous driving systems. Built on the Unreal Engine, it offers realistic rendering of urban layouts, dynamic actors (such as vehicles and pedestrians), and diverse environmental conditions, including varying weather and lighting scenarios. A key advantage of CARLA is its flexibility in configuring sensor suites and environmental parameters, allowing researchers to simulate a wide range of scenarios and test the robustness of autonomous agents under different conditions. This includes the ability to model complex traffic situations, such as intersections with multiple agents, adherence to traffic rules, and unexpected events like pedestrians crossing the road or sudden obstacles. Such capabilities are crucial for developing and evaluating RL algorithms intended for dynamic and unpredictable environments.

2.3 Model Predictive Control

Pérez-Dattari et al. [2022] introduce a novel motion planning framework that combines visual inputs with human feedback to generate socially compliant driving behaviours. The framework consists of two main components: a data-driven policy that interprets visual information to produce high-level driving decisions, and a local trajectory optimisation method that ensures the safe execution of these decisions. By employing Interactive Imitation Learning, the authors jointly train the policy and a Model Predictive Controller (MPC), resulting in driving behaviours that closely resemble human actions. The approach is validated in realistic simulated urban scenarios, demonstrating improvements in safety—evidenced by a reduction in collisions—and data efficiency compared to prior methods. This work highlights the potential of integrating human-in-the-loop learning with traditional control techniques to enhance the performance and social acceptability of autonomous vehicles.

Schwenzer et al. [2021] provide an extensive review of MPC, emphasising its practical applications

across various engineering domains. The paper delves into the theoretical foundations of MPC, its historical evolution, and the challenges associated with its implementation, particularly the computational demands. By highlighting real-world applications in areas such as power electronics, process industries, and mechanical systems, the authors demonstrate the versatility and effectiveness of MPC in handling complex control tasks. This review serves as a valuable resource for understanding the current state of MPC and its potential future developments.

Ramezani et al. [2024] present an innovative control framework that combines PPO, a RL technique, with MPC to manage the dynamics of uncertain floating platforms in zero-gravity conditions. Conducted within the Zero-G Lab at the University of Luxembourg, their approach leverages the predictive capabilities of MPC to guide the PPO learning process, resulting in a system that adapts effectively to unmodeled dynamics and external disturbances. Experimental results demonstrate that this integrated PPO-MPC method achieves faster convergence and superior stabilisation performance compared to PPO alone, highlighting its potential to enhance the reliability of autonomous systems in space exploration contexts.

This thesis adopts gradient-based RL approaches, which optimise policies by computing gradients of expected reward with respect to policy parameters. Two popular algorithms are used: PPO, known for its balance between performance and stability, and SAC, which incorporates entropy maximisation to encourage exploration. These methods are evaluated in dynamic environments to assess their adaptability to evolving conditions.

2.4 Chapter conclusion

This chapter reviews the foundational and contemporary research that underpins the current study, with a focus on algorithms, simulation environments, and MPC. Foundational texts such as Sutton and Barto's work established the core principles of RL, while more recent advances, including those in Deep RL, have extended these ideas to complex, high-dimensional problems. The comparative analysis of PPO and SAC algorithms has revealed their respective strengths in dynamic environments, providing justification for their selection in this thesis.

The review of simulation environments has underscored the importance of designing adaptive reward functions and flexible goal setting mechanisms to reflect real-world challenges. Based on previous work, this study adopts a streamlined reward structure that prioritises collision avoidance, and introduces a dynamic goal-update strategy to foster long-term planning and adaptability.

Central to this simulation strategy is the use of CARLA, an open-source high-fidelity simulator designed for autonomous driving research. CARLA enables safe and repeatable testing in richly detailed 3D urban environments, complete with dynamic elements such as traffic, pedestrians, and varying weather conditions. Its flexibility in sensor configuration and scenario generation makes it an ideal platform for evaluating the adaptability and performance of RL algorithms under realistic conditions.

Finally, the exploration of MPC highlights its relevance for integrating predictive control into autonomous systems, particularly when combined with RL. Hybrid approaches, such as PPO-MPC, demonstrate the potential for improved stability and adaptability in uncertain or changing conditions.

Together, these insights provide a robust theoretical and methodological foundation for the experiments conducted in subsequent chapters. By situating this thesis within the broader research landscape, this chapter highlights how the current work seeks to advance the field through the

application and evaluation of modern RL algorithms in complex, dynamic 3D environments.

3 Experimental setup

To get answers to these questions, I will use CARLA to train a self-driving car. The car should be able to drive from point A (start) to a random point B (end) without colliding with anything. The experimental design consists of a single task executed in multiple configurations. The objective is for an autonomous agent to navigate a vehicle in a straight line across an intersection, covering a distance of up to 20 metres. The experiment is carried out three times, each with a different control strategy to evaluate performance under varied conditions.

- Proximal Policy optimisation (PPO)
- PPO combined with Model Predictive Control (PPO+MPC)
- Soft Actor-Critic (SAC)

In each trial, the agent starts from a fixed position. Execute a sequence of actions with the aim of reaching a designated goal point while avoiding collisions. After each attempt, the agent updates its policy based on the outcome, either successful completion or failure, and then restarts the task using the learnt policy.
The goal is initially placed three metres from the starting position. After each successful episode, the goal is incrementally moved farther away, 1 metre at a time, until it reaches a final distance of twenty metres. This progressive increase in distance allows the agent to gradually learn more complex behaviours as the difficulty of the task increases.
The experiment is carried out within a simulated intersection environment. This setting is selected for its dynamic nature; traffic can approach from multiple directions, making it one of the most complex and variable scenarios in autonomous driving. As such, it provides a meaningful context for evaluating the adaptability and robustness of different learning algorithms.

3.1 Reinforcement learning

3.1.1 State space and actions

The RL environment is responsible for processing agent actions and returning updated states. In this work, the environment is implemented using the CARLA simulator, a high-fidelity platform for autonomous driving research. The simulated vehicle in CARLA is equipped with multiple sensors, many of which produce high-dimensional image data, resulting in a large state space due to the per-pixel information content. Fortunately, CARLA also provides access to sensors that output lower-dimensional structured data, which can simplify state representation and reduce computational complexity.
For this implementation, the agent's state is derived from a radar sensor, yielding a 14-dimensional continuous state vector. The first two elements represent the relative position of the goal and the heading angle towards it. The remaining 12 values encode information about the six closest objects, including their relative distances and velocities with respect to the sensor.
The action space is continuous and two-dimensional, with both values restricted to the interval $[-1, 1]$. The first value controls the steering angle, while the second regulates the throttle or braking input.

3.1.2 Rewards

As discussed previously, in RL, the environment provides rewards associated with specific states and actions. In the context of our environment, we evaluate multiple reward structures to guide the agent's behaviour effectively.
The initial reward structure assigns a negative reward for collisions and a positive reward for successfully reaching the goal. However, this formulation proved insufficient, as the agent lacked a clear incentive to progress toward the goal, often opting to avoid movement altogether to minimise the risk of collisions.To ensure that training episodes would complete within a reasonable time frame, we imposed a time limit of 100,000 timestamps per episode.
To address this issue, we introduced a distance-based penalty, in which the agent incurs a negative reward proportional to its distance from the goal at each timestamp. This modification effectively encourages the agent to minimise its distance from the goal, thereby promoting goal-directed behaviour. However, this adjustment introduced a new challenge. Since every non-terminal state receives a negative reward and the number of such states increases with the distance to the goal, the total cumulative reward collected during an episode decreases as the goal is placed farther away. This is undesirable as we aim for a reward structure that promotes consistent performance improvement over time. The agent's behaviour also regressed at times to remaining stationary, as waiting for the episode to time out resulted in a higher cumulative reward than actively attempting to reach the goal and risking a collision.
To mitigate this, we considered two alternatives: (1) providing incremental rewards for each unit of distance reduced between the agent and the goal, or (2) scaling the terminal reward based on the total distance from the start to the goal. We adopted the latter approach. Specifically, upon reaching the goal, the agent receives a reward equal to 1,000 times the total distance travelled. The final reward function is therefore defined as follows:

- **Goal reached:** Reward = 1,000 $\times$ distance
- **Collision:** Reward = $-10{,}000$
- **Each non-terminal state:** Reward = $-$distance to goal

This revised structure effectively balances incentives for goal-reaching behaviour while preserving reward scalability across varying episode lengths.

3.2 PPO

PPO employs an Actor-Critic (AC) architecture (Konda and Tsitsiklis [1999]), where a neural network simultaneously estimates both the policy (actor) and the value function (critic). The actor network outputs a probability distribution over possible actions, from which an action is sampled and executed by the agent. After the agent interacts with the environment and observes the resulting new state, this information is fed back onto PPO to update the policy and determine subsequent actions. The shared Actor-Critic network consists of an input layer, followed by three hidden layers of sizes 128, 256, and 128 respectively, and finally separate output layers for the actor and the critic.

To get an action, the AC network returns a mean and a standard deviation. Then, PPO makes a normal distribution from which it samples an action. The standard deviation term determines the exploration-exploitation trade-off.
In our PPO implementation, the clipping parameter ϵ was set to 0.2, which effectively controls the maximum allowable policy update per training step to ensure stability. The learning rate used to optimise the actor and critic networks was 1×10^{-4}, balancing convergence speed and training stability. For each policy update, the algorithm performs 15 epochs on the collected batch of experience, with a batch size of 4096. The discount factor γ, which determines the importance of future rewards, was set at 0.99, while the Generalised Advantage Estimation parameter λ was chosen as 0.95 to balance bias and variance in the calculation of the advantage. Unlike some policy gradient methods, exploration in this approach is driven solely by the learnt standard deviation of the Gaussian action distribution of the policy, without using an entropy regularisation term. Furthermore, we are using an Adam optimiser to train the AC network.

3.3 SAC

In this implementation of SAC, both the actor and critic networks are modeled as fully connected neural networks, each comprising three hidden layers with 128 units per layer and employing ReLU activation functions. To address overestimation bias, two critic networks are maintained concurrently, following the twin Q-learning framework. Corresponding target networks are updated using a soft update mechanism with a smoothing coefficient of $\tau = 0.005$.
The actor and critic losses follow the standard SAC formulation. The entropy temperature coefficient α is fixed at 0.2 throughout training, rather than being learned, as is done in some variants. This simplification allows for more controlled comparisons and improved stability in training.
Experiences are stored in a replay buffer with a capacity of $100,000$ transitions, from which mini-batches of size 4096 are sampled during training. For environments with multi-dimensional action spaces, log-probabilities are summed across action dimensions when computing the entropy term. The most notable deviation from the standard SAC formulation is the use of a fixed entropy coefficient, as opposed to adapting α dynamically. This modification was made to reduce algorithmic complexity and promote consistent learning behaviour.
The following hyperparameters were used throughout all experiments: a learning rate of 1×10^{-4}, a discount factor $\gamma = 0.99$, a soft update coefficient $\tau = 0.005$, a fixed entropy temperature $\alpha = 0.2$, and a batch size of 4096.

3.4 MPC

MPC is a decision-making strategy that plans a sequence of actions over a fixed time horizon by predicting future states. At each time step, MPC simulates how the system will evolve under various action sequences and evaluates these simulated trajectories based on a predefined objective or cost function. Then, the best trajectory is selected and the first action is executed before repeating the process at the next time step. This approach allows the system to continuously adapt to changes in the environment.
In this study, we use an MPC horizon of five, meaning each candidate trajectory consists of five state-action pairs. For each decision step, three such trajectories are simulated, and the best one is selected based on a cost function that evaluates criteria such as proximity to obstacles and path

efficiency to determine the immediate control action. An initial configuration using a horizon of 15 and 10 candidate trajectories was also tested, but this led to significant computational overhead and reduced responsiveness. To improve real-time performance and enable more reactive behaviour, the horizon and number of trajectories were reduced accordingly.
Importantly, these trajectories are simulated internally, without directly interacting with the CARLA environment during the decision-making process. Instead, vehicle dynamics, such as velocity and position, are estimated on the basis of a simplified internal model. Although this introduces a degree of inaccuracy, it more closely mirrors real-world decision-making, where a vehicle cannot "replay" multiple options in the environment before choosing the best. Rather, it must rely on internal predictions to guide its actions.
This setup highlights the practical trade-off between accuracy and real-time feasibility. Although it may lack the fidelity of simulating every trajectory in the full environment, it allows for fast, reactive decision-making, aligning with how autonomous systems must operate under real-world constraints.

3.5 Chapter conclusion

This chapter outlined the experimental framework developed to evaluate the performance of different control strategies for autonomous vehicle navigation in a complex intersection scenario. By leveraging the CARLA simulator, we created a controlled yet dynamic environment to test the ability of RL agents to safely and efficiently drive from a fixed starting point to an incrementally more distant goal.
We implemented and tested three control strategies: PPO, PPO enhanced with Model Predictive Control (PPO+MPC), and SAC. Each strategy was evaluated through repeated trials, with the goal distance increasing after each successful episode to progressively challenge the agent's capabilities. The experiment emphasised safe and goal-directed behaviour, with a carefully designed reward structure that balanced the need for exploration, progress, and collision avoidance.
The PPO algorithm utilised an Actor-Critic architecture with clipped updates to ensure stability during policy improvement, although SAC incorporated entropy maximisation to encourage robust exploration. The hybrid PPO+MPC approach introduced an anticipatory planning component, enabling more reactive decision-making without relying solely on policy learning. The experimental parameters, such as network architectures, learning rates, discount factors, and batch sizes, were tuned to ensure comparability and training efficiency across all approaches.
Together, these methodologies enabled a thorough investigation into the trade-offs between learning-based and planning-based approaches. The setup provides a robust platform for assessing generalisation, adaptability, and safety in autonomous driving systems - key components in real-world deployment scenarios. The results of these experiments will offer insight into the strengths and limitations of each method, setting the stage for the comparative performance analysis presented in the following chapter.

4 Results

Having established the methodology, we now present the results of our experiments. These findings highlight the key results observed during the testing and form the basis for a subsequent discussion.

4.1 PPO

The initial experiment was conducted using only the PPO agent. In this setup, the agent received an environmental state, selected an action accordingly, executed the action, and received a new state in return. This cycle was repeated continuously throughout each episode.
As illustrated in Figure 1, the reward increases rapidly at the beginning of each episode, which is consistent with typical RL dynamics, where the agent quickly learns basic behaviours that yield positive outcomes. Following this initial phase, the reward generally stabilises near the maximum attainable value, as expected. However, after approximately episode 100, some runs exhibit a decline in performance. This can be attributed to the agent either becoming too slow or experiencing frequent collisions, both of which lead to reduced rewards. It is also noteworthy that the graph terminates around episode 200. The reasons for this vary between runs: in some cases, the agent began to idle and prolong the episode duration unnecessarily, while in others, the simulation experienced a crash. Despite these inconsistencies, most runs reached at least 200 episodes, and this cut-off sufficiently captures the relevant performance trends. Therefore, the graph was truncated at this point for clarity and consistency.

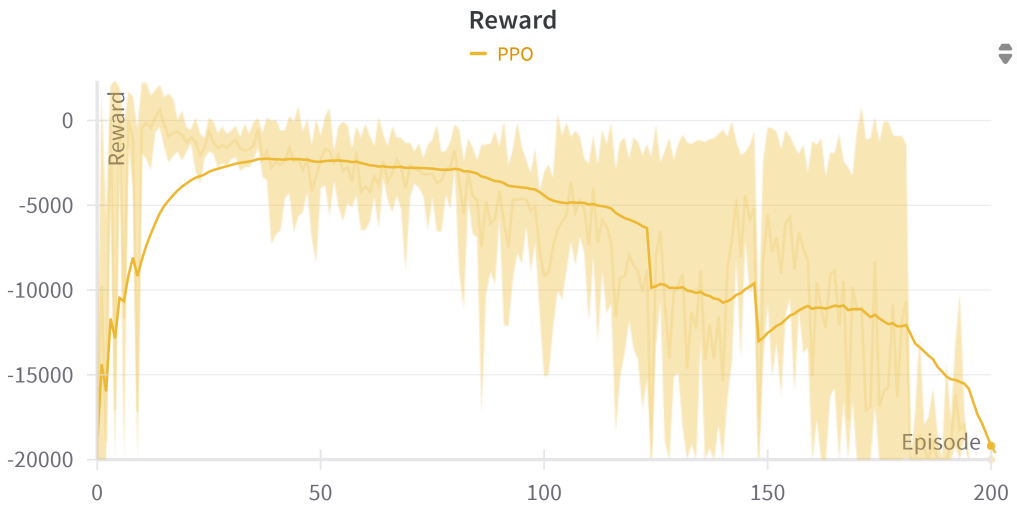

Figure 1: Episode-wise reward during training using the PPO algorithm. The plot shows rapid initial improvement followed by increased variance and eventual decline in performance, suggesting convergence instability or policy degradation over time.

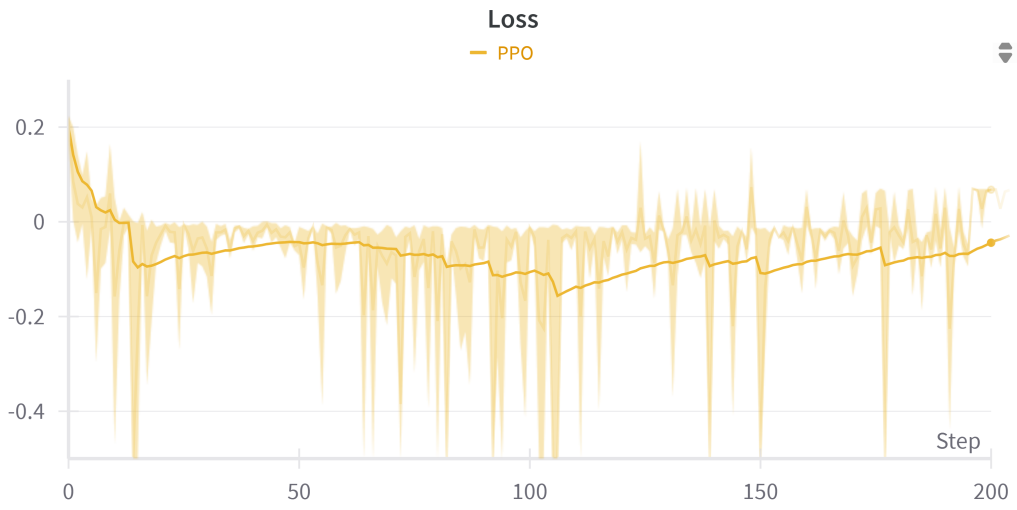

Figure 2: Training loss across steps for the PPO algorithm. The loss shows early stabilisation with intermittent spikes and fluctuations, indicating periods of unstable policy updates or transitions during learning.

Figure 2 shows that the loss function remains relatively stable throughout training, especially when compared to the variability observed in the reward trends. The mean loss centres around approximately -0.1, with occasional negative outliers. This pattern aligns with expected behaviour in RL, where the loss typically begins at a higher value and gradually stabilises as the agent improves its policy. In this case, the stabilisation occurs around -0.1, indicating convergence in the policy updates.
This stability in the loss function complements the reward dynamics discussed in Figure 1. While rewards fluctuate more visibly due to environmental interactions and episode-specific events (e.g., collisions or idle behaviour), the relatively consistent loss values suggest that the underlying policy optimisation is proceeding as intended. The combination of rising rewards and stabilising loss reflects effective learning and policy refinement during the training process.

4.2 PPO + MPC

During initial trials, the stand-alone PPO algorithm failed to produce satisfactory results, often struggling to navigate effectively to the goal. To address this, PPO was combined with MPC in an effort to enhance stability and performance. Although early results of the integrated approach mirrored the same ineffective behaviour, subsequent debugging led to noticeable improvements. The agent began to consistently reach its destination, initially moving cautiously, but eventually achieving a more reasonable speed.
However, a technical issue related to the simulation environment emerged during the training. The agent spawn point was located on an active roadway, which occasionally caused nearby vehicles to collide with the agent immediately after spawning. Although the system was designed to wait for the spawn area to clear, high-speed vehicles approaching from a short distance sometimes caused unavoidable collisions. These incidents led to disproportionately large penalties—of 10,000—recorded

on episodes of minimal duration, thus skewing some training results. To mitigate this issue, we excluded all episodes with fewer than 10 timestamps from the training data.

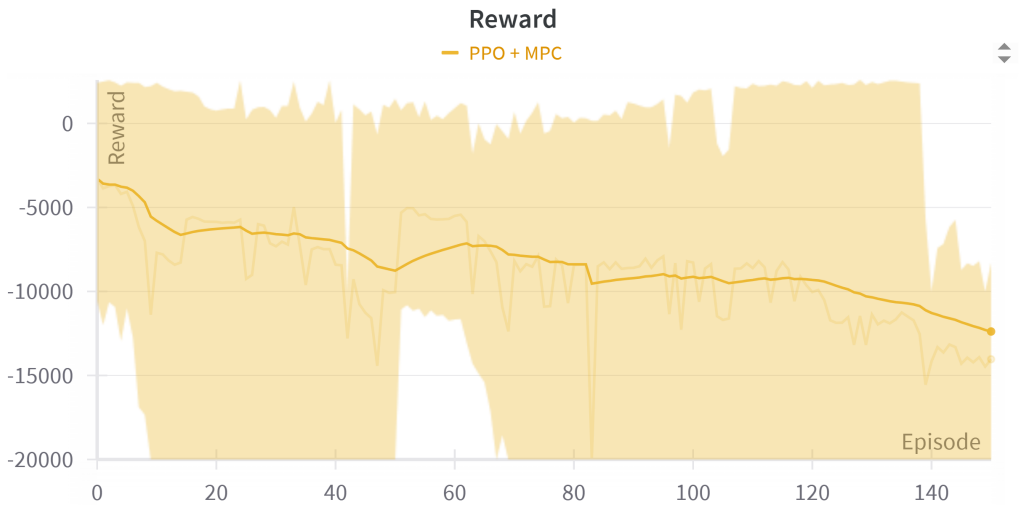

Figure 3: Episode-wise reward during training using the PPO + MPC algorithm. The shaded region indicates the variance over multiple runs. A general decreasing trend in reward is observed, highlighting performance instability.

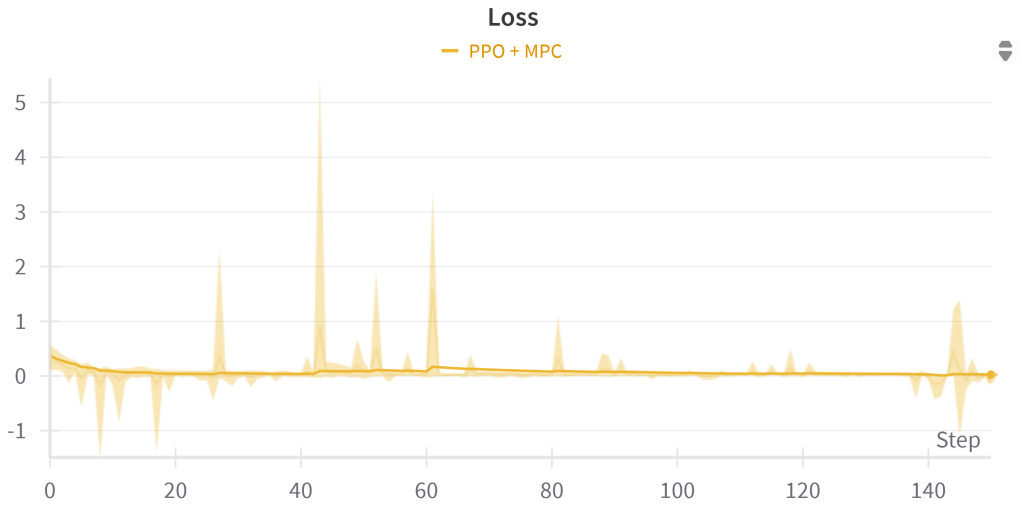

Figure 4: Training loss over time for the PPO + MPC algorithm. While the loss generally stabilises, occasional spikes are visible, indicating moments of high prediction error or learning adjustment.

As illustrated in Figure 4, the loss decreases progressively towards zero over time, reflecting the expected training behaviour. In contrast, the results shown in Figure 3 diverge from the anticipated outcome. The mean reward initially starts at a relatively high value, likely because the goal is easily

attainable, making exploratory actions more susceptible to suboptimal performance. Additionally, the agent frequently reverts to its initial plan, as evidenced by the oscillatory pattern in the reward curve.
Although the mean reward exhibits a periodic pattern, the overall trend is downward, which is not desirable. This decline can be attributed to the increasing distance the vehicle must travel, resulting in more time steps to reach the goal. In our environment, the agent incurs a penalty at each time step, proportional to its distance from the goal, to encourage goal-directed movement. However, when the initial distance is larger, the cumulative penalty increases accordingly. To mitigate this effect, we adjusted the reward for reaching the goal to be proportional to the travel distance, but this modification proved insufficient.

4.3 SAC

In contrast to PPO and PPO + MPC, which required significantly longer to complete each episode and, consequently, each run, the SAC algorithm demonstrated much faster performance. In most cases, SAC was able to complete the driving task and update its policy in under two minutes. This efficiency explains why the reward curve in Figure 6 terminates after approximately 1000 episodes, compared to 150–200 episodes in the PPO-based experiments. Additionally, due to the shorter runtime per episode, it was possible to conduct a greater number of experimental runs with SAC, resulting in a denser and more compact graph, especially when contrasted with Figure 3.

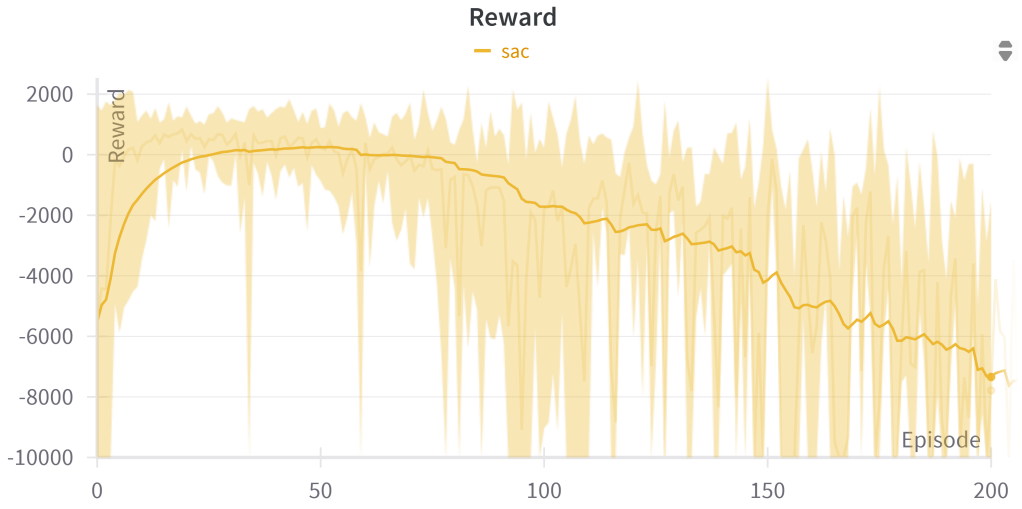

Figure 5: Zoomed-in view of the SAC training reward curve (episodes 0–200) shown in Figure 6. The agent initially learns effective behaviours, with a rapid increase in reward followed by a plateau and a gradual decline. The early training dynamics highlight a strong learning phase before long-term degradation sets in.

Figure 5 reveals reward dynamics similar to those previously observed in Figure 1. The reward begins at a low value, increases rapidly as the agent learns effective behaviours, and then plateaus. However, around episode 100, a gradual decline in performance is observed, likely due to slower driving speeds or increased inefficiency. After approximately episode 700, the reward appears to

stabilise again, at around −75,000. It stabilises even more from 850, because than the goal stopped moving. This plateau is not indicative of performance degradation per se, but rather a consequence of the predefined time limit imposed on each episode. In many of these later episodes, the SAC agent continued to drive toward the goal but did so too slowly to reach it within the allowed time, resulting in no reward being assigned. Without such a time constraint, it is reasonable to expect that the reward curve would have continued to decline at the gradual rate observed between episodes 200 and 600.

Figure 6: Full episode-wise reward curve during training using the SAC algorithm. After initial improvement, the reward steadily declines, indicating a deterioration in the agent's ability to consistently achieve the goal. This trend suggests potential issues with long-term policy stability.

A notable difference between the PPO-based experiments and the SAC experiment lies in the way the goal was incrementally moved. In the PPO-based setups, the goal position was shifted by 1 metre every 50 episodes until it reached a distance of 20 metres from the agent. In contrast, the SAC experiment employed a finer-grained progression, moving the goal by 0.1 metre every 5 episodes—also culminating in a final distance of 20 metres. In both cases, the initial goal was placed 3 metres from the agent, implying that it would require a total of 850 episodes to cover the full 20-metre distance.

4.4 Chapter conclusion

Given that neither of the PPO-based experiments extended to 850 episodes, it is evident that they did not complete the full task. In contrast, SAC successfully covered the entire distance within the duration of its experiment. While this may not be immediately apparent from Figure 6, closer inspection reveals that toward the end of the graph, the reward curve becomes relatively flat with minimal downward deviation. Instead, occasional upward deviations suggest that the agent was sometimes able to reach the goal slightly more efficiently than in previous episodes. This consistent performance toward the end supports the conclusion that SAC was able to complete the task successfully, reaching the 20-metre goal within the allotted time.

As observed in both the PPO-based and SAC experiments, the reward curves exhibit typical RL behaviour during the initial phase, stabilising after a period of rapid improvement. However, a noticeable change occurs around episode 100. At this point in training, the agent is required to travel approximately 5 metres to reach the goal, which yields a reward of 5,000. However, the penalty incurred per timestamp has also increased to 5 (up from an initial value of 3), and the longer distance requires more timestamps to complete.

As a result, the cumulative penalties begin to outweigh the reward, particularly when the agent is slow or inefficient. This imbalance likely causes the overall reward to decrease, leading to the downward trend observed in the reward graphs. This effect is most clearly illustrated in the SAC experiment, where the decline in performance is more pronounced after episode 100, corresponding to the increased difficulty and higher penalty accumulation.

5 Conclusion

In this study, we explored various RL algorithms and environments, focusing on a comparative analysis of PPO, PPO combined with MPC and SAC. Among these, SAC demonstrated the most promising performance, successfully reaching the final goal within the set time constraints. This outcome suggests that SAC was the best suited for the tasks evaluated in these experiments. We also aimed to assess performance in terms of collision avoidance; however, this proved difficult due to the incomplete execution of the PPO-based experiments.
The experiments highlighted how different reward structures can significantly influence learning outcomes. As discussed in chapter three, multiple reward configurations were tested, with the selected structure offering the most potential. Nevertheless, the trends observed in the reward graphs raise important questions: Was the underlying concept correct but the implementation flawed, or was the reward design itself fundamentally misguided? Future research could address this by testing alternative structures, such as applying a constant per-timestamp penalty, introducing exponentially scaled rewards for goal completion, or providing incremental rewards as the agent approaches the goal.
Additionally, we examined the specific influence of MPC. However, based on the available results, no definitive conclusion can be drawn about whether MPC enhances or hinders PPO's performance. Both the reward and loss graphs were limited in scope, and given PPO's relatively poor performance on its own, it remains unclear how much MPC contributed, positively or negatively.
The central research question posed was: Can a reinforcement learning agent be trained to navigate autonomously in CARLA based on collision avoidance?
In summary, the answer is yes, an RL agent can indeed be trained to navigate autonomously in CARLA with an emphasis on collision avoidance. However, the more complex question is how this can best be achieved. While SAC showed promise, the overall findings were inconclusive and leave room for substantial improvement. Further experimentation, particularly with refined reward structures and extended training durations, could offer deeper insights. It would also be valuable to revisit PPO under these improved conditions to better assess its capabilities.
Although the experiments did not deliver the results we hoped for, they delivered hope for better results.

5.1 Limitations

Several limitations were encountered during the course of the experiments. The first major constraint was the experimental setup itself. All experiments were performed on a single personal computer, which was also required for other tasks. This limited the available computational resources and significantly slowed the training process compared to what could have been achieved using cloud-based infrastructure. In addition, time constraints played a critical role. Due to these limitations, a decision was made not to fully complete the PPO-based experiments and instead proceed with the partial results available at the time.
Another key limitation was related to the CARLA simulator. Although CARLA proved to be a powerful and flexible tool for simulating autonomous driving environments - particularly due to its compatibility with Python, which was used for the experiments - it also presented several challenges. The simulator is resource-intensive, requiring a substantial amount of memory and computing power. Moreover, its complex installation process made it difficult to set up the environment on

alternative machines, thereby restricting the ability to parallelise or relocate the experiments.

5.2 Future work

Although some suggestions for future work have already been mentioned, this section elaborates on several promising directions for further research. The primary recommendation is to explore more deeply how a RL agent can be effectively trained in the context of autonomous navigation. The agent's performance is largely influenced by the design of the reward structure, with a secondary influence from the choice of hyperparameters. Given that the reward structure used in this study did not produce consistently effective results, it would be logical to prioritise this area for future research.

Several alternative approaches have already been proposed, such as applying a fixed per-timestamp penalty rather than one based on distance, introducing exponentially scaled rewards for goal completion, or awarding incremental rewards as the agent moves closer to the goal. These methods could provide more stable learning signals and better incentivise efficient behaviour.

In addition, it would be valuable to investigate the discrepancy in performance between PPO and SAC. Understanding why PPO underperformed while SAC showed more promising results could offer key insights into the suitability of different algorithms for this type of task. Finally, a compelling line of inquiry would be to evaluate whether integrating MPC with SAC—as opposed to PPO—can further enhance performance. Such experiments could help isolate the contribution of MPC and better understand its interaction with various RL algorithms.

References

Alexey Dosovitskiy, German Ros, Felipe Codevilla, Antonio Lopez, and Vladlen Koltun. CARLA: An open urban driving simulator. In *Proceedings of the 1st Annual Conference on Robot Learning*, pages 1–16, 2017.

Vijay Konda and John Tsitsiklis. Actor-critic algorithms. In S. Solla, T. Leen, and K. Müller, editors, *Advances in Neural Information Processing Systems*, volume 12. MIT Press, 1999. URL https://proceedings.neurips.cc/paper_files/paper/1999/file/6449f44a102fde848669bdd9eb6b76fa-Paper.pdf.

Yuxi Li. Deep reinforcement learning: An overview. *CoRR*, abs/1701.07274, 2017. URL http://arxiv.org/abs/1701.07274.

Abu Jafar Md Muzahid, Syafiq Fauzi Kamarulzaman, and Md Arafatur Rahman. Comparison of ppo and sac algorithms towards decision making strategies for collision avoidance among multiple autonomous vehicles. In *2021 International Conference on Software Engineering Computer Systems and 4th International Conference on Computational Science and Information Management (ICSECS-ICOCSIM)*, pages 200–205, 2021. doi: 10.1109/ICSECS52883.2021.00043.

Rodrigo Pérez-Dattari, Bruno Brito, Oscar de Groot, Jens Kober, and Javier Alonso-Mora. Visually-guided motion planning for autonomous driving from interactive demonstrations. *Engineering Applications of Artificial Intelligence*, 116:105277, 2022. ISSN 0952-1976. doi: https://doi.org/10.1016/j.engappai.2022.105277. URL https://www.sciencedirect.com/science/article/pii/S0952197622003323.

Mahya Ramezani, M. Amin Alandihallaj, and Andreas M. Hein. Ppo-based dynamic control of uncertain floating platforms in the zero-g environment, 2024. URL https://arxiv.org/abs/2407.03224.

M. Schwenzer, M. Ay, T. Bergs, and D. Abel. Review on model predictive control: An engineering perspective. *The International Journal of Advanced Manufacturing Technology*, 117 (5–6):1327–1349, 2021. doi: 10.1007/s00170-021-07682-3. URL https://doi.org/10.1007/s00170-021-07682-3.

Sharma. *Sharma, S. (2020). SAC-RL: Continuous Control of Wheeled Mobile Robot for Navigation in a Dynamic Environment (thesis).* PhD thesis, 2020.

Richard S. Sutton and Andrew G. Barto. *Reinforcement learning an introduction Richard S. Sutton, Andrew G. Barto.* A Bradford Book, 1998.