Міністерство освіти і науки України

Національний університет «Києво-Могилянська академія»

Факультет інформатики

Кафедра мультимедійних систем

Кваліфікаційна робота

освітній ступінь – бакалавр

на тему: «**ENERGY CONSERVATION IN AUTONOMOUS AGENTS USING REINFORCEMENT LEARNING**»

Виконав: студент 4-го року навчання,

Освітньої програми «Комп'ютерні науки», 122

Беймук Володимир Олегович

Керівник: Кузьменко Д.О.,

Старший викладач

Рецензент: Швай Н.О.,

Доцент, к. ф-мат. наук

Кваліфікаційна робота захищена
з оцінкою _____

Секретар ЕК _____
«____» _____ 20____ р.

Київ – 2025

Міністерство освіти і науки України
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «КИЄВО-МОГИЛЯНСЬКА АКАДЕМІЯ»
Кафедра мультимедійних систем факультету інформатики

ЗАТВЕРДЖУЮ
зав. кафедри мультимедійних систем,
доцент, к.ф-м.н.
Жежерун О.П.
_____
(підпис)
«_____» _____ 2025 р.

ІНДИВІДУАЛЬНЕ ЗАВДАННЯ

на кваліфікаційну роботу

студенту Беймуку Володимиру Олеговичу

факультету інформатики 4 курсу бакалаврської програми

ТЕМА: Energy Conservation in Autonomous Agents using Reinforcement Learning

Зміст ТЧ до кваліфікаційної роботи:

Індивідуальне завдання

Календарний план

Анотація

Зміст

Вступ

Попередні дослідження

Методи

Експерименти

Результати

Висновки

Примітки

Список літератури

Дата видачі «___» _____ 2025 р. Керівник Кузьменко Д.О.

_____
(підпис)

Завдання отримав _____
(підпис)

Тема: Energy Conservation in Autonomous Agents using Reinforcement Learning

Календарний план виконання роботи:

№	Назва етапу кваліфікаційної роботи	Термін виконання	Примітка
1.	Визначення теми кваліфікаційної роботи.	05.10.2024	
2.	Отримання завдання на кваліфікаційну роботу.	30.10.2024	
3.	Огляд необхідних джерел.	02.02.2025	
4.	Тестування наявних рішень.	07.03.2025	
5.	Вибір технологій для модифікації функції винагороди.	19.03.2025	
6.	Експерименти з модифікацією функцій винагороди.	30.04.2025	
7.	Написання текстової частини.	21.05.2025	
8.	Передзахист кваліфікаційної роботи	26.05.2025	
9.	Захист кваліфікаційної роботи.	05.06.2025	

Студент Беймук В.О.

Керівник Кузьменко Д.О.

«___» _____ 2025 р.

АНОТАЦІЯ

Метою кваліфікаційної роботи є дослідження можливостей алгоритмів навчання з підкріпленням для формування стратегій автономного водіння з урахуванням компромісу між енергоефективністю та швидкістю проходження траси. Основну увагу приділено модифікації функції винагороди шляхом додавання штрафу за витрату пального.

Проведено аналіз сучасних підходів у сфері автономного водіння та перегонів, а також визначено основні труднощі, пов’язані з багатокритеріальною оптимізацією у реалістичних середовищах для перегонів.

Описано реалізацію завдання з використанням алгоритму Soft Actor-Critic у середовищі Assetto Corsa. Розглянуто вплив різних рівнів штрафу за витрату пального на навчання агента, а також досліджено, як змінюються стратегія водіння, витрата пального та час проходження кола залежно від обраних параметрів винагороди. Окрему увагу приділено аналізу основних факторів, що впливають на витрату пального в симуляторі, таких як прискорення, оберти двигуна, передача та амплітуда кермового кута.

Ключові слова: автономне водіння, навчання з підкріпленням, компроміс швидкість-ефективнсть, Soft Actor-Critic, Reinforcement Learning, енергоефективність, симуляція перегонів.

TABLE OF CONTENTS

1 INTRODUCTION

As autonomous driving technology continues to advance, it holds significant potential to revolutionize future mobility. The potential benefits of this technology range from reducing traffic congestion to minimizing traffic accidents caused by human error [1, 36]. However, developing effective autonomous agents involves complex challenges related to perception, planning, control, and decision-making in unpredictable and dynamic environments [2].

Within this broader field, autonomous racing has emerged as a particularly demanding and insightful research area [3]. Much like how traditional motorsport has historically driven innovation in technology for commercial vehicles, autonomous racing pushes the limits of autonomous driving systems by requiring vehicles to operate at the limits of their capabilities [4]. This makes it a challenging testbed for developing high-performance, safe, and efficient algorithms [5]. The primary objective in autonomous racing is typically to complete a given track as fast as possible, which often requires precise inputs at the vehicle's physical limits [4].

In this context, Reinforcement Learning (RL) has become a popular machine learning approach for training autonomous agents through trial and error by interacting with their environment. This methodology has been successfully applied in autonomous driving [16, 17]. It has shown great success in such scenarios due to its ability to handle unprocessed high-dimensional data inputs. [10,11]. RL has been applied to various autonomous driving tasks, including motion planning [12], highway driving [13], and autonomous racing [4, 6-9, 14], including maneuvers such as overtaking [15].

While the focus of autonomous agents is typically on speed and performance [4, 6-9], energy efficiency is usually not prioritized despite its growing societal and

environmental impact. A few recent papers have successfully applied RL for optimizing fuel economy in autonomous agents by learning efficient velocity profiles [18]. However, there remains a gap in the literature regarding approaches that explicitly penalize energy usage in autonomous driving contexts.

Our work aims to address this gap by exploring the intersection of autonomous racing, reinforcement learning, and energy conservation. Specifically, we investigate how incorporating a fuel consumption penalty into the reward function affects the resulting strategies of a reinforcement learning agent in a high-fidelity racing environment. Our goal is to incentivize the agent to find a balance between speed and energy efficiency, thereby forcing it to learn non-trivial trade-offs. Our key hypothesis is that by shaping the reward function to penalize fuel use in conjunction with lap time performance, the agent can be encouraged to adopt energy-efficient driving strategies without significantly compromising lap time.

This work aims to contribute to a broader understanding of how reinforcement learning can adapt for multi-objective optimization in realistic driving scenarios. By utilizing a high-fidelity simulator, we provide a realistic environment to test these ideas and explore their outcomes. Our results suggest that even mild fuel penalties can enable a more fuel-efficient driving behavior, often without compromising performance. In some configurations, agents not only matched but slightly outperformed the baseline in both lap time and fuel usage. Our work highlights the importance of reward design when applying reinforcement learning to multi-objective control tasks, such as autonomous racing.

2 RELATED WORK

2.1 Introduction to Autonomous Driving and Reinforcement Learning

2.1.1 Overview of Autonomous Driving Research

Autonomous driving systems have undergone significant development in recent years. Traditionally, those systems follow a perception-planning-control pipeline, which involves three main stages:

- Perception: Sensor data is processed to perceive the environment
- Planning: The system plans a general path or a concrete maneuver
- Control: Low-level control commands are executed

This paradigm has been widely applied in both industry and research, particularly in the development of self-driving cars and autonomous racing strategies. It is convenient to implement and use since each part can be designed and tested independently. However, it also means the system must make assumptions at each state, leading to slow and complex environments [19].

While simulation plays a critical role in training autonomous agents – it still comes with its challenges when transferring models to the real world. This is often referred to as the sim-to-real gap. Simulators struggle to accurately replicate real-world conditions, which often include sensor noise, changing weather, varying road textures, and vehicle conditions. These mismatches mean that a policy that performs well in a virtual environment might fail when tested on an actual vehicle [1, 20]. As noted in sim2real studies, even minor differences can result in significant performance drops during real-world testing. This highlights the need for more adaptable policies that can generalize beyond the specific conditions in which they were trained.

2.1.2 Reinforcement Learning in Control Tasks

More recently, a growing body of research has emerged concerning end-to-end systems, particularly those utilizing imitation learning and reinforcement learning [17], which offer promising approaches to solving these problems. Reinforcement Learning (RL), in particular, has been widely used in control tasks due to its ability to learn complex behaviors through trial and error. Compared to traditional rule-based approaches, RL can adapt to dynamic and unpredictable environments.

RL is a framework for learning optimal decision-making policies through continuous interactions with an environment. Formally, RL problems can be modeled as a Markov Decision Process (MDP), defined by a tuple (S, A, P, R, y), where

- S is the set of states that the environment can take at a given step
- A is the set of actions that the agent can take at a given step
- $P(s'|s, a)$ is the probability of transitioning to state s' from state s after taking action a.
- $R(s, a)$ is the reward function
- $y \in [0,1]$ is the discount factor for future rewards

The most important assumption of MDP is the Markov property, which states that the future state depends only on the current state and action, not on the whole history of previous states and actions.

Using MDP, reinforcement learning algorithms, learn a policy and/or a value function that the agent uses to choose an action that leads to the highest reward.

In continuous control settings, such as driving, robotic arm manipulation, or locomotion, RL has been successfully applied to train agents that optimize long-term objectives, including safety, efficiency, and speed [2, 10, 12, 21, 22].

In the context of autonomous driving, RL is more often defined as a Partially Observable Markov Decision Process (POMDP), meaning that the agent cannot directly observe the whole state of the environment. This is often used for more realistic learning conditions since, in the real world, the true state of the environment (e.g., complete road layout, sensor state) is not fully observable, and agents rely on partial and noisy observations, which is why, in PODMP, the agent receives observations that provide partial information about the underlying state. A POMDP is defined by $(S, A, P, R, \Omega, \mathrm{O}, \mathrm{y})$, where:

- S, A, P, R, y are as in an DMP
- Ω is the set of possible observations
- $\mathrm{O}(o|s, a)$ is the observation function: the probability of observing o after taking action a and reaching state s)

2.1.3 Model-Free vs. Model-Based RL

In RL, algorithms are often categorized into model-free and model-based approaches, depending on how they interact with the environment and learn their value function or policy from it. Figure 1 schematically demonstrates the difference between these two approaches.

Model-based RL methods attempt to learn a model of how the environment behaves. They predict the next state and the reward of the environment, given the current state and the action taken. The model uses its prediction to simulate possible future scenarios and choose actions that lead to better outcomes. An example of such an approach is the Deep Q-Network (DQN) [23].

In contrast, model-free RL does not perform the modeling step. Instead, the agent interacts directly with the environment and learns a policy or a value function through trial and error. Some examples of such include algorithms like PPO [24] and SAC [25].

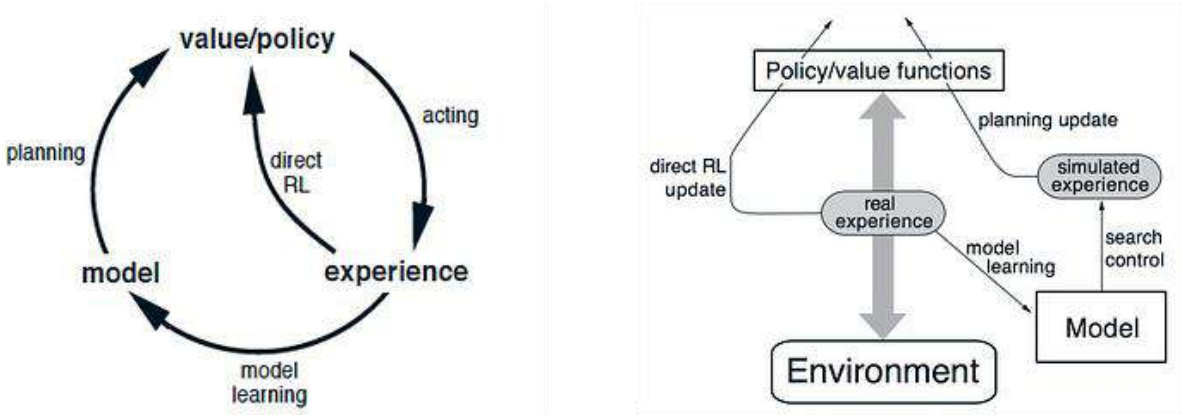


Figure 1. The difference between model-free (left) and model-based (right) RL algorithms [26].

Model-free methods are generally more robust and simpler to implement because they do not rely on maintaining a model of the environment. This is particularly useful in complex or difficult-to-model environments. However, model-free methods require large amounts of interaction with the environment, which makes them less sample-efficient and slower to train. In contrast, model-based methods can be significantly more sample-efficient, as they can simulate additional experiences using the learned model of the environment. However, the performance of model-based methods greatly depends on the accuracy of the learned model. In highly complex and unpredictable environments, even small errors can accumulate, causing unstable and suboptimal policies.

Due to the high complexity of autonomous racing environments, we opted for a model-free approach. In our case, modeling the simulator would likely be time-consuming and prone to errors. A model-free method allows us to learn directly through interactions with the environment. Such an approach offers greater robustness and reliability at the cost of increased training time. Additionally, model-free algorithms like SAC have been successfully applied in similar tasks in the autonomous racing domain [4], which gives us confidence about their suitability and effectiveness for our use case.

2.2 Soft Actor-Critic (SAC) and Related Algorithms

2.2.1 Overview of SAC

Soft Actor-Critic is a model-free, off-policy reinforcement learning algorithm introduced by Haarnoja et al. and designed specifically for continuous control tasks [25]. This algorithm learns both a policy (the “actor”) and a value function (the “critic”) simultaneously. SAC is unique due to its use of entropy regularization, which makes the policy less likely to converge in local optima. Entropy regularization works by rewarding the agent for taking less certain and more exploratory actions.

SAC is also off-policy, meaning it can learn from previously collected data rather than only using data from current policy. This makes it more sample-efficient, which is especially useful when working with complex environments that require running a large number of episodes.

For driving and racing tasks, SAC is a good fit for several reasons. First, it supports continuous action spaces, which is important for inputs like throttle, brake, and steering. Second, entropy regularization helps to avoid suboptimal lap times and provides a necessary balance between exploration and exploitation. Finally, its ability to learn from past experiences means we can make better use of each training episode [17].

2.2.2 SAC in Autonomous Driving

Soft Actor-Critic (SAC) has been widely used in autonomous driving research due to its robustness, stability, and suitability for continuous control tasks. SAC has been applied to various tasks, including high-speed racing [4], overtaking [15], and highway driving [13] in realistic simulators such as Gran Turismo, TORCS, CARLA, and Assetto Corsa.

In recent studies, deep RL agents trained using SAC-based algorithms have successfully outperformed professional human drivers, achieving super-human precision and optimizing tactical maneuvering [27]. Furthermore, those agents can execute autonomous overtaking maneuvers at the limit of car dynamics, outperforming both traditional model-based controllers and experienced human drivers.

The usage of SAC in these applications is justified by its ability to handle continuous action spaces, off-policy nature, and entropy regularization, as described in the previous section. However, SAC also has limitations. Like other model-free methods, it requires a large amount of interaction data, making it computationally expensive to train, especially in complex simulators. Additionally, SAC does not explicitly model uncertainty or maintain a belief about the state, which can be problematic in partially observable environments like autonomous driving (due to occluded sensors or delayed telemetry). It may also struggle with multi-objective optimization unless provided with carefully designed reward functions.

Despite these challenges, SAC has proven to be one of the most effective reinforcement learning algorithms for autonomous driving, particularly in high-speed racing environments.

2.3 Simulation Environments for RL in Driving

RL research in autonomous driving has historically moved towards standardized environments to ease reproducibility, benchmarking, and experiments. OpenAI Gym was one of the earliest frameworks to achieve this [28]. It provides intuitive interfaces for a wide range of simple environments with additional tools for designing custom environments using Python and PyTorch.

There have been a variety of other simulators specifically designed for robotic tasks, such as Gazebo or AirSim. They offer greater fidelity by modeling the whole 3D environments, realistic physics and can add sensor data from LiDAR, radar, and cameras. One such simulator, CARLA, has been specifically designed for autonomous driving [29]. It has become a popular open-source option for research in this area due to its realistic physics and great accessibility.

Other popular platforms like TORCS and SUMO target different use cases. TORCS is tailored toward racing and control tasks, but it lacks support for many features, such as path planning and weather conditions [31]. SUMO, on the other hand, is a 2D traffic flow simulator used mostly for macroscopic traffic research rather than low-level control [30]. Both SUMO and Torcs support only a 2D view of the scene.

In our research, we use the OpenAI Gym interface designed by [4] that integrates a popular commercially available simulator. This choice allows us to combine the flexibility and simplicity of the Gym API with the realism of a professional racing simulator. Using the Gym interface makes it easier to integrate modern reinforcement learning algorithms and design custom reward functions.

2.4 Prior Work in RL for Racing Tasks

The research in autonomous racing mainly focuses on minimizing lap time, which contrasts with general autonomous driving, where a lot of research focuses on safety [6, 34]. RL is increasingly used in autonomous driving to learn complex policies from experience and feedback from the environment due to the difficulties of designing optimal rule-based or modular control strategies in dynamic and unpredictable settings.

Research in this area focuses on using RL to optimize for speed, lap time, and precision. The goal in racing is often to complete a track as fast as possible, which

means minimizing lap time [6, 7]. RL algorithms are considered effective for learning to drive from telemetry data [6]. Studies have explored various RL algorithms for this purpose, including Deep Deterministic Policy Gradient (DDPG) [9, 10, 14, 33], Proximal Policy Optimization (PPO) [9, 32], Deep Q-Network (DQN) [9], Asynchronous Actor-Critic (A3C) [16], and Soft Actor-Critic (SAC) [34], and LSTM [35]. These methods aim to learn policies that output continuous control actions like steering angle and throttle/brake inputs.

Some work incorporates techniques like a look-ahead of the curve (LAC) approach, which has been shown to improve performance in self-racing scenarios [6]. Methods have also been developed to reduce the space of continuous actions, for example, by discouraging simultaneous throttle and brake inputs [6].

While the primary focus in autonomous racing is speed, some general autonomous driving studies using RL incorporate additional factors. For example, a reward function for lane following and overtaking includes an “efficiency or overtaking factor” [35].

Performance evaluation in autonomous driving is often measured in terms of distance travelled, car speed, and smoothness/stability [11].

RL is also being used to reduce fuel consumption in driving tasks. One approach used by Kim et al. [18] is to train a neural network to act as a velocity block generator, which predicts the most fuel-efficient speeds based on rod information and remaining trip time.

2.5 AssettoCorsaGym Platform

2.5.1 Architecture and Features

AssettoCorsaGym offers a Gym-compatible interface combined with the Assetto Corsa simulator [4]. This integration allows reinforcement learning algorithms to

interact with a realistic driving environment using standard Gym methods. The simulator's telemetry and control systems are accessed through a custom plugin architecture. Sensor data is transmitted to the Gym interface (e.g., car velocity, acceleration, tire friction, steering angle) at 50Hz and exposed through Python bindings. Control commands are sent using a virtual joystick interface (vJoy), which contains throttle, brake, and steering inputs.

The platform supports traditional controllers like Model Predictive Control (MPC), as well as RL algorithms such as Soft Actor-Critic (SAC) and RL from Demonstrations (RLfD). Human driving data was collected using the Assetto Corsa Telemetry Interface (ACTI) and stored in the MoTeC format. The system includes logging, ROS2 integration, and compatibility with both Linux and Windows.

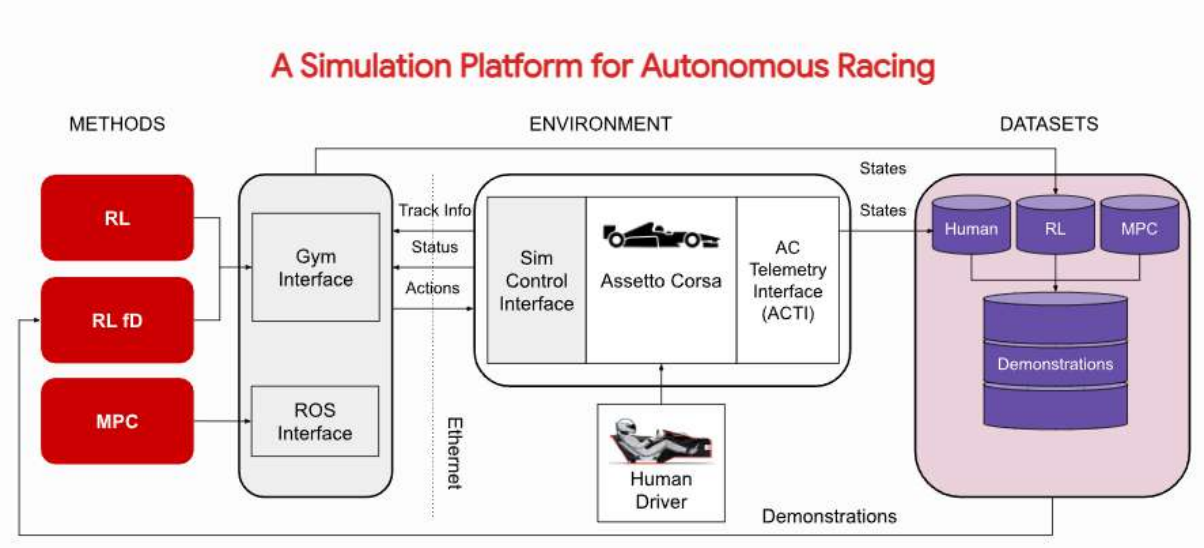


Figure 2. The architecture of the AssettoCorsaGym platform. The diagram shows the simulator and the connection to different control methods (RL, RLfD, MPC) via the Gym and ROS interfaces. Human driver demonstrations are collected separately and logged through the telemetry interface to create a dataset from human data for training and benchmarking. The repository supports multiple methods for autonomous racing – built-in AI, MPC, SAC, and TD-MPC2 [4]

2.5.2 Use in Research

The original AssettoCorsaGym paper evaluates SAC, MPC, and TD-MPC2 [4]. Performance is compared against Assetto Corsa's built-in AI and human drivers, including a few experts. SAC and TD-MPC2 were trained from scratch and with human demonstrations [4].

Experiments were conducted using three cars of varying complexity – a lightweight rear-wheel drive car, a GT series car, and an open-wheel Formula 3 series car. The tracks' difficulty also varied from simple oval circuits to complex maps with high-speed sections and complex technical turns [4].

The results show that SAC and TD-MPC2 were able to match or exceed the lap times of professional human drivers in most scenarios. TD-MPC2 demonstrated strong generalization and few-shot learning, making it promising for transfer learning across different tracks. MPC, while generally slower, remained more stable and, therefore, valuable for tasks that require reliability and safety [4].

Despite its strengths, the platform has its limitations. One drawback is that it only runs in real-time, which slows down training for computationally expensive model-free RL algorithms like SAC. Another is the use of a predefined reference path in the reward function, which limits its flexibility for multi-agent learning, where agents might be required to deviate from the standard paths to perform overtaking maneuvers [4]. Additionally, for the purposes of our research, the platform did not initially transfer nor use information about fuel consumption in any way, making it difficult to evaluate energy-efficient driving strategies without further modifications.

2.6 Summary and Research Gap

Recent work in autonomous racing has demonstrated the effectiveness of RL agents in practice. Algorithms such as SAC have shown strong results in high-fidelity environments, often outperforming expert human drivers.

Despite these advances, several limitations remain. First, energy efficiency is rarely addressed in such research. Most works focus on minimizing lap time or improving specific factors like control precision, but little consideration is given to fuel consumption and efficiency trade-offs. This leaves a gap in understanding how reinforcement learning policies behave when specific constraints are introduced, such as fuel limits.

Second, while realistic simulators are used more often, they are still underutilized in RL research. Many studies continue to rely on simplified or abstract environments that fail to capture the complexity of actual vehicle dynamics. As a result, the policies trained in those simplified settings struggle to generalize to more realistic conditions.

Third, existing approaches tend to overlook partial observability in their problem formulation. In practice, driving agents rarely have access to full state information, such as exact fuel level, tire degradation, or road friction. However, most RL algorithms, especially when applied to racing environments, assume full observability.

Our work aims to address these gaps by extending the Soft Actor-Critic to incorporate fuel efficiency objectives while using a realistic simulator. Our approach uses a high-fidelity environment compatible with the Gym interface, allowing us to define custom reward functions that account for fuel usage without including the information about the exact fuel level in the observation space. This setup allows us to explore how reinforcement learning agents adapt when required to make decisions under realistic resource constraints. In addition, we

aim to compare the behavior of trained agents to human driving strategies in the scenarios when fuel conservation is required. Through this, we hope to better understand how RL can learn to balance performance and energy efficiency in complex control tasks.

3 METHODS

3.1 RL Environment

For our experiments, we used the AssettoCorsaGym repository [4], which provides a solid foundation for reinforcement learning research in realistic racing simulations. It integrates the Assetto Corsa racing simulator with the OpenAI Gym framework.

The repository implements a communication layer between the Gym interface and the simulation engine. Specifically, the Sim Control Interface retrieves telemetry data from the Assetto Corsa API and transmits the current state of the environment to the Gym interface directly to the RL agent.

3.2 RL Policy

We used the Soft Actor-Critic (SAC) algorithm for our experiments due to its strong performance in the benchmarks and its general robustness and sample efficiency in control environments [4]. SAC is especially great for difficult tasks such as autonomous racing due to its ability to develop optimal policies that balance speed, control, and long-term planning [25]. This is enabled by entropy regularization, which encourages exploration and prevents premature convergence.

3.3 Cars and Tracks Dataset

The setup included two racetracks – Track A and Track B (Figure 3), and two vehicle models: a lightweight Formula 3 series car (F3) and a heavier GT3 series car (GT3). This dataset was selected because it aligns with those used in the original AssettoCorsaGym dataset, which allows for simplifying direct comparison with the benchmark results.

The chosen tracks present distinct challenges: Track A offers a balanced layout with both technical turns and high-speed sections, making it suitable for evaluating general driving performance, while Track B includes complex chicanes and long straights, making it the most demanding track in the original dataset [4]. Table 1 compares the main characteristics of the tracks.

The two chosen vehicles differ significantly in dynamics, which allowed us to test the model's performance across different driving strategies. The F3 series car is a lightweight, high-downforce formula car that is nimble and relatively easy to control, while the GT3 is a heavier, high-power touring vehicle that requires more conservative handling, especially during complex cornering.

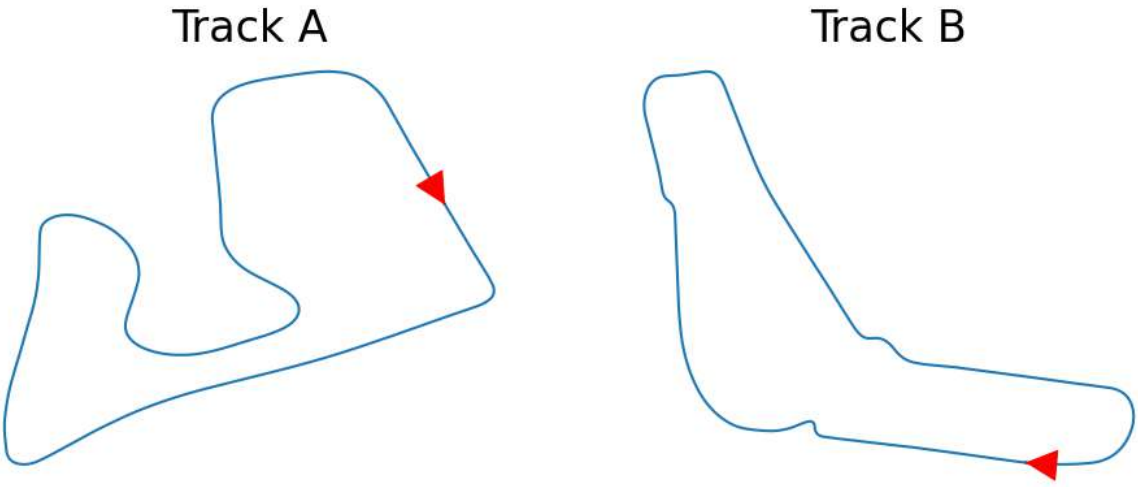


Figure 3. The visual representation of two tracks that were chosen for the training and evaluation of the model. The tracks were chosen from the Assetto Corsa

simulation environment. The red triangle indicates the start position and direction of the map.

Track	Length (km)	Turns
A	4.318	10
B	5.793	11

Table 1. Comparison of Track A and B in terms of length and the number of turns. Track A represents a balanced circuit that is great for evaluating the agent's general performance. Track B is a more complex and longer map, with precise turns and extended high-speed sections.

3.4 Fuel setup

By default, the AssettoCorsaGym interface did not expose fuel consumption data to the Gym environment [4]. To address this, we extended the simulators communication layer by modifying the Sim Control Interface to include fuel status information in the "State" packet. This fuel data was then incorporated into the agent's simulation state and used for reward computation.

To simulate a more realistic driving scenario and test the agent's ability to develop fuel-efficient behavior under partial observability, we decided not to include fuel consumption into the agent's observation space. Although fuel usage influences the reward function, the agent must infer its impact indirectly through its interactions. This design encourages the agent to develop energy-efficient driving strategies without direct supervision.

To calculate the reward, we computed the immediate change in fuel level between simulation steps (denoted as delta_fuel). Additionally, we logged the remaining fuel level at each step, along with fuel usage statistics per episode, including:

- **BestLapFuel**: Fuel used during the lap with the lowest consumption.

- **LapNo_i_Fuel**: Fuel consumed during lap *i*.

Refueling is handled automatically at the end of each episode, by resetting the car's fuel tank. During training, we set the fuel consumption rate to 0.1 to avoid premature depletion. However, for evaluation and visualization, we recalculated fuel consumption using the simulator's standard rate.

The initial fuel levels were set to 30 liters for the GT3 series car and 22 liters for the F3 series car. This corresponds to their typical usage profiles.

3.5 Environment Fixes and Modifications

All experiments were conducted using a forked version of the AssettoCorsaGym repository [4]. This fork was not continuously synced with the upstream due to frequent breaking changes introduced in newer commits. However, since the reinforcement learning algorithms remained unchanged during this period, the modifications introduced by the authors did not affect training performance. We opted to maintain a stable version for reproducibility and consistency.

During initial testing, we encountered a bug in the environment where the agent would stop responding to steering inputs after a random number of simulation steps. This behavior significantly hindered training, as it required manual restarting of the environment each time it occurred.

After encountering the issue, we submitted a GitHub issue and asked the authors for feedback. The root cause was traced to the vJoy driver, which emulates control inputs for the simulator. Occasionally, vJoy would stop transmitting actions to the simulation at a random simulation step. To resolve it, we modified the environment to explicitly load and release the vJoy DLL at the beginning and end of each episode, rather than only at the beginning of training. After implementing the fix, we submitted it as a GitHub pull request to the AssettoCorsaGym repository.

3.6 Fuel consumption calculation and analysis

We were using the built-in fuel consumption metrics calculated in the Assetto Corsa simulator. Since the exact fuel consumption formula was hidden in the actual game files and encrypted, we did not have access to it. Nevertheless, we wanted to measure what is the nature of this metric and what components affect it. This would show whether the real-world fuel saving strategies would be efficient in the simulator.

To measure this correlation, initially, we had to address the problem of corresponding the fuel changes to the agent's actions. Our current delta fuel at step x was computed as the difference between the fuel at step x and step x-1, i.e. $\Delta fuel(x) = fuel(x-1) - fuel(x)$, and so, our current delta fuel would correlate the fuel change at step x with the action taken at step x-1.

However, for our analysis, we needed to establish a direct mapping between the action taken at a specific time step and its immediate effect on fuel consumption. That is, if the agent applies acceleration at step x, we wanted the delta fuel associated with that action to appear at the same step. To achieve this, we calculated:

$$\Delta fuel(x) = fuel(x) - fuel(x+1)$$

To then measure the effect of different metrics on immediate fuel consumption, we used partial correlation between delta fuel and metrics from the simulator, such as acceleration status, car roll and yaw, steering angle amplitude, braking and so on.

We computed Pearson correlation coefficients for different metrics, while including every other metric in covariates, to measure how much each metric separately affects fuel consumption. Our results are summarized in Table 2.

Feature	Partial correlation (r)	P-value	Confidence Interval (95%)
Acceleration	0.94	0.00	[0.93, 0.94]
Engine RPM	0.59	5.95×10^{-218}	[0.56, 0.62]
Braking	-0.29	5.84×10^{-47}	[-0.33, -0.26]
Steer Angle Amplitude	0.20	4.50×10^{-23}	[0.16, 0.24]
Speed	0.08	1.16×10^{-4}	[0.04, 0.12]
Roll	0.07	2.31×10^{-3}	[0.03, 0.11]
Yaw	-0.01	6.12×10^{-1}	[-0.05, 0.03]

Table 2. Partial correlation coefficients between selected features and immediate fuel consumption (delta_fuel). Pearson coefficients are reported for each feature. Positive coefficients indicate the increase of fuel use with the increase of the feature. Negative coefficients show that fuel usage decreases when the feature is increased. The results confirm that throttle and engine RPM show by far the strongest correlation with fuel consumption. Additionally, the amplitude of the braking input and steering wheel angle show a moderate correlation, while speed, roll, and yaw show a weak correlation.

We then tested the relationship between acceleration status and fuel consumption, since this acceleration status was the most significant metric according to Pearson coefficients. As shown in Figure 4, the resulting scatter plot reveals a nonlinear correlation, with delta fuel increasing with acceleration but displaying variance across the range.

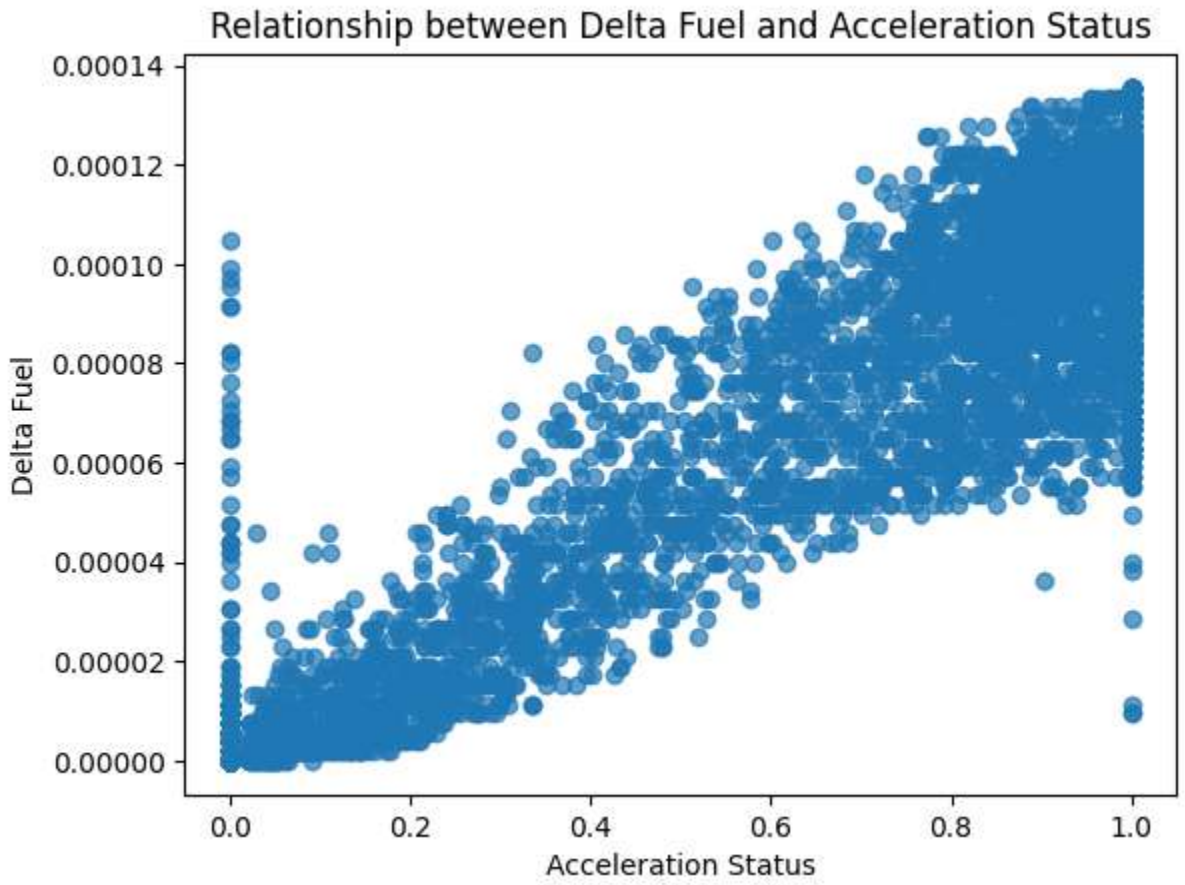


Figure 4. Scatter plot of acceleration status vs. delta fuel, showing a nonlinear relationship.

3.7 Reward function modifications

As mentioned, our experiments required modifications to the reward function provided in AssettoCorsaGym, to account for fuel consumption and encourage the agent to develop fuel-efficient driving policies.

The default reward function provided in AssettoCorsaGym is based on the car's velocity and penalizes deviation from the optimal driving line. It is computed as:

$$r = v \cdot (1 - a \cdot d)$$

where v is the car's current speed, d is the L2 distance from the optimal path (as determined by the simulator), and a is a configurable penalty coefficient [4].

To encourage fuel-efficient driving behavior, we extended this reward function by incorporating a penalty for fuel consumption. Specifically, we added a term based on the immediate fuel change since the previous timestep (delta_fuel), denoted as f. The resulting reward function is defined as:

$$r = v \cdot (1 - a \cdot d) - b \cdot f$$

where b is a tunable coefficient that controls the strength of the fuel consumption penalty. Higher values of b increase the penalty and incentivize more efficient driving strategies. Our implementation allows the $b \cdot f$ term to be toggled on or off via configuration.

Notably, as mentioned previously fuel consumption was not included in the agent's observation space and was only used to influence the reward. As a result, the agent could not directly observe the fuel level or consumption rate during training and had to learn fuel-efficient behavior through implicit feedback. This partial observability setup was intended to simulate more realistic conditions and encourage new strategies.

We conducted experiments with multiple values of b to achieve an approximate reduction in reward of 2%, 5%, 10%, and 20%, relative to the original formulation. Since fuel consumption varies across different vehicles, depending on their internal parameters (e.g., weight, aerodynamics, gear ratios, engine type), we calculated the appropriate b coefficient individually for each car. The resulting values are summarized in the table below:

Car model	**2% penalty**	**5% penalty**	**10% penalty**	**20% penalty**
GT3 series	139	360	719	1439
F3 series	249	623	1240	2480

Table 3. Values of the b coefficient used in the reward function. Since fuel consumption depends on the intrinsic parameters of the car, the coefficients need

to be recalculated for every car. The penalty percentage is relative to the original formulation of the reward function from AssettoCorsaGym [4]. The penalty levels were higher for the F3 series car, since on average it consumed less fuel per one simulation step. Therefore, it required higher values for the *b* coefficient to match the desired percentage of the original reward.

To keep the implementation simple and computationally efficient, the reward function only considers the immediate fuel consumption at each step. It might be worth exploring the effects of incorporating a short history of past fuel usage into the state space, or penalizing fuel-related factors directly such as sustained high RPM or aggressive acceleration.

The simulation also does not support manual gear shifting, which limits the agent's ability to optimize fuel usage via advanced gearbox management.

All training runs were performed on an RTX 3060 laptop GPU and an NVIDIA T4 GPU hosted remotely via an Azure Virtual Machine. On average, it took approximately two days to complete 500 training episodes.

4 EXPERIMENTS

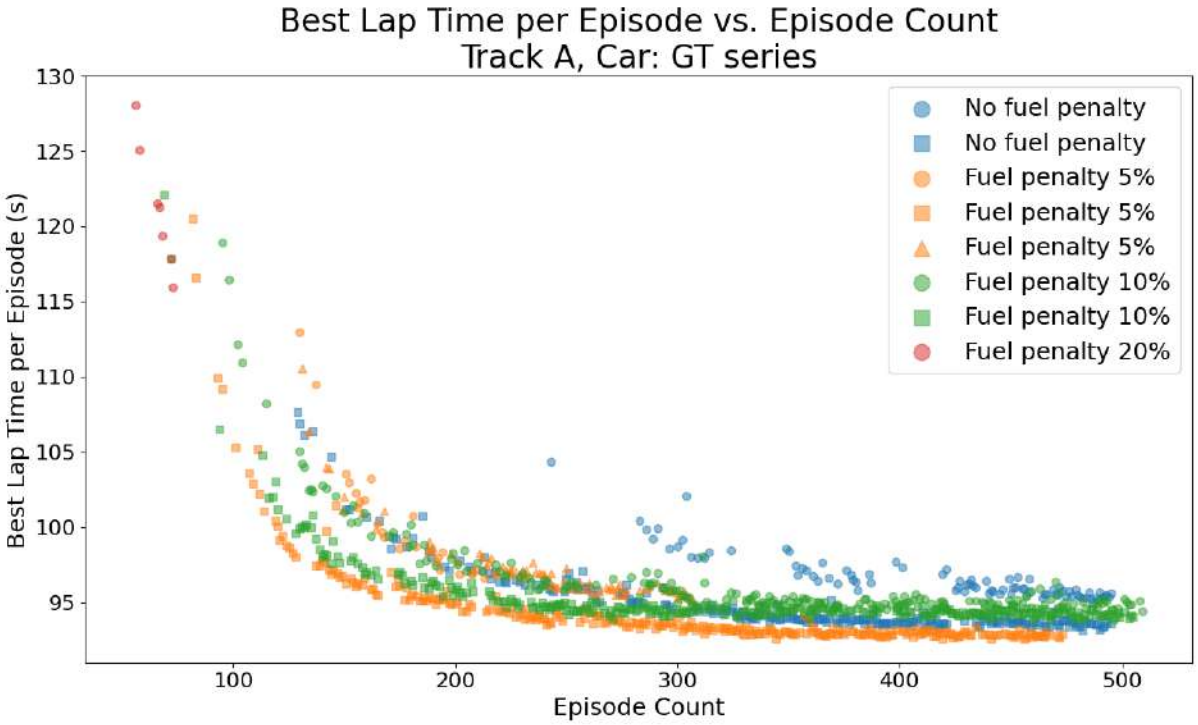
We evaluate how introducing a fuel penalty into the reward function affects the training and performance of the reinforcement learning agent. Specifically, we experiment with four penalty strengths – corresponding to approximate reward reductions of 2%, 5%, 10%, and 20% – to investigate their impact on driving behavior, fuel consumption per lap measured in liters, and lap time performance.

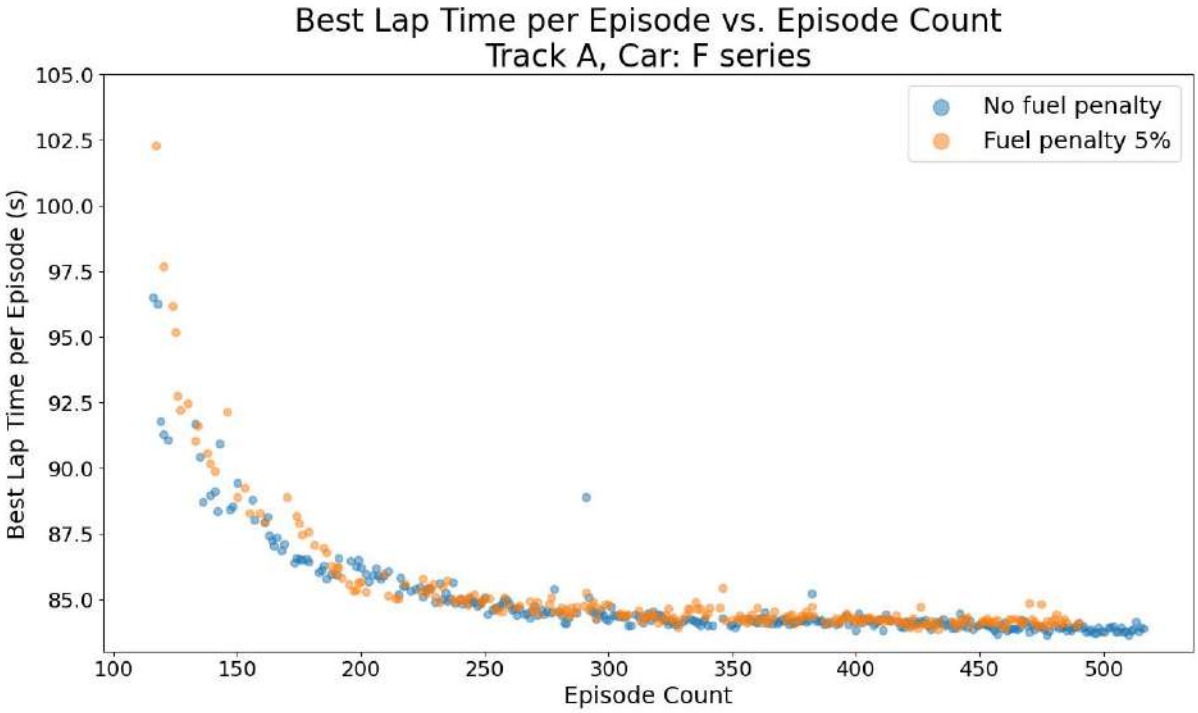
4.1 Lap Times Trends and Learning Curves During Training

We investigated how different levels of fuel consumption penalty affect the learning of the RL agent. As shown in Figure 5, models trained with low to moderate fuel penalties (2%-5%) often achieved better lap times during early training compared to baseline models without any penalty.

This effect was most noticeable on Track A, and with the GT3 car, where agents with moderate penalties consistently outperformed the baseline during the first 200 training episodes. However, this advantage did not always persist later in the training. In several cases, especially with higher penalty coefficients (e.g., 10%-20%), models converged to suboptimal policies that significantly prioritized fuel savings over speed. Most notably, we have observed that the model trained with a 20% penalty failed to reach competitive lap times during training, meaning that this level of penalty may limit the agent's ability to learn faster strategies.

Moreover, this trend did not generalize to Track B, where models with fuel penalties performed worse than the baseline. In particular, models trained with penalties of 5% or higher consistently failed to complete a single lap during training. We attribute this to Track B's challenging layout, where the environment with the heavy fuel penalty was too constrained for the model to learn efficiently.


Best Lap Time per Episode vs. Episode Count
Track A, Car: GT series
Best Lap Time per Episode (s)
Episode Count
No fuel penalty
No fuel penalty
Fuel penalty 5%
Fuel penalty 5%
Fuel penalty 5%
Fuel penalty 10%
Fuel penalty 10%
Fuel penalty 20%


Best Lap Time per Episode vs. Episode Count
Track A, Car: F series
Best Lap Time per Episode (s)
Episode Count
No fuel penalty
Fuel penalty 5%

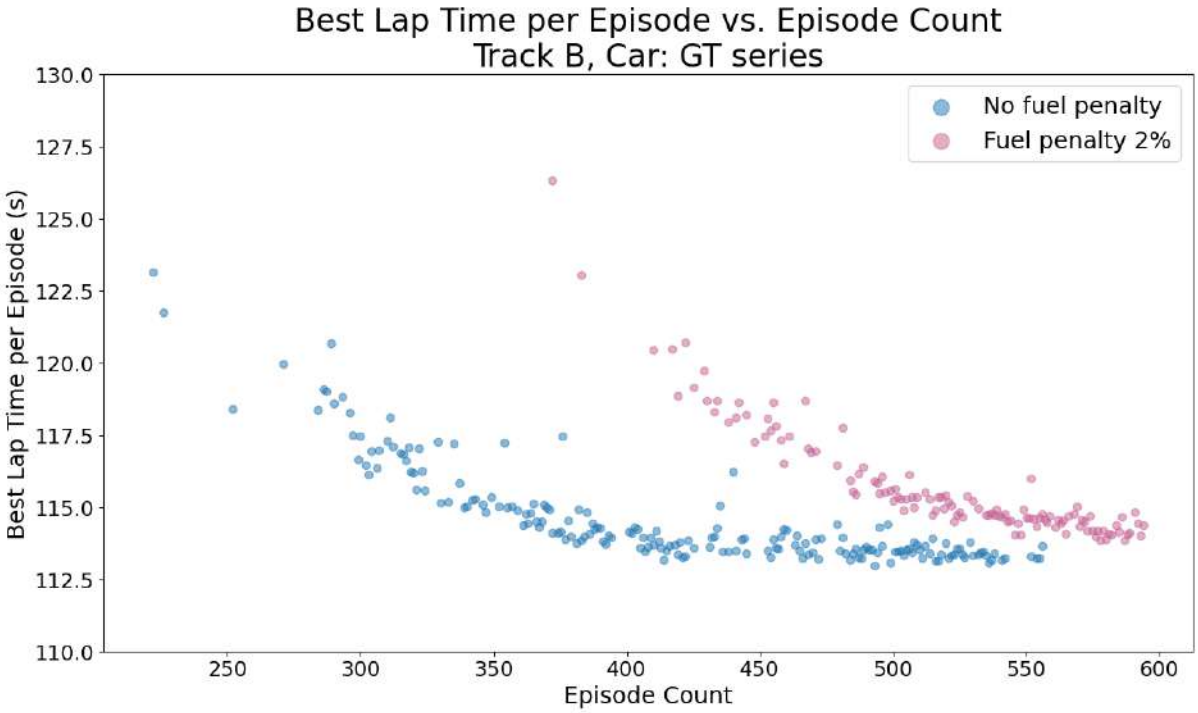


Figure 5. Best lap times per episode across different training simulations with various levels of fuel consumption penalty. Each subplot shows the training progression for a specific car and track combination. Colors indicate reward function variants with different fuel penalty coefficients. Shapes correspond to different random seeds. Models with light penalties often converge faster than baseline, while heavy penalties often lead to suboptimal lap times.

To compare performance across all combinations of cars, tracks, and reward functions, we summarize the best observed lap times and corresponding fuel consumption values in Table 4.

Track A, Car: GT3 series			
Fuel Penalty	**Random Seed**	**Best Lap Time (s) ↓**	**Fuel/Lap for Best Lap ↓**
0%	0	94.97	1.88
	1	93.13	1.80
5%	0	96.59	1.88
	1	**92.57**	1.59

	2	95.09	1.47
10%	1	94.35	1.44
	2	93.80	1.43
20%	1	115.93	**1.01**

Track A, Car: F3 series			
Fuel Penalty	**Random Seed**	**Best Lap Time (s) ↓**	**Fuel/Lap for Best Lap ↓**
0%	0	**83.62**	0.97
5%	0	83.86	**0.90**

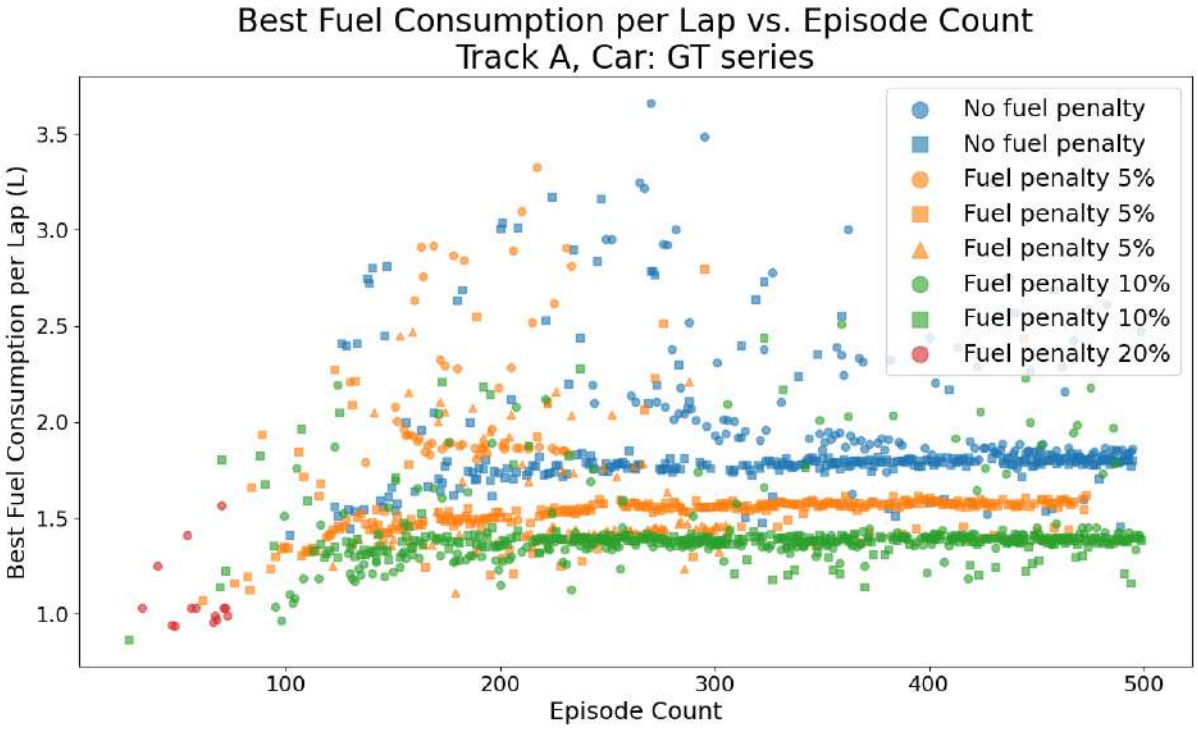
Track B, Car: GT3 series			
Fuel Penalty	**Random Seed**	**Best Lap Time (s) ↓**	**Fuel/Lap for Best Lap ↓**
0%	0	**112.99**	2.45
2%	0	113.85	**2.39**
5%	0	DNF	DNF
10%	0	DNF	DNF

Table 4. Comparison of best lap times and corresponding fuel consumption per lap across different fuel penalty levels, vehicle models (GT3 and F3), and tracks (Track A and Track B). Fuel penalty levels are relative to the mean reward obtained by the agent per simulation step. Some configurations were evaluated using multiple random seeds. Results show the impact of fuel penalty on fuel efficiency and lap time performance. “DNF” values indicate that the agent failed to complete a valid lap.

4.2 Fuel Consumption Trends During Training

Figure 6 shows the changes of best fuel consumption rates per lap across training episodes for different reward function configurations. Across all training configs, introducing a fuel penalty consistently led to lower fuel usage over time, showing that the agent learned to adapt its driving style for better fuel efficiency.

On Track A, both the GT3 and the F3 agents trained with fuel penalties were able to consistently achieve lower fuel consumption compared to the no-penalty baseline. For Track B, while the general trend remained, the impact of the penalty was less visible.



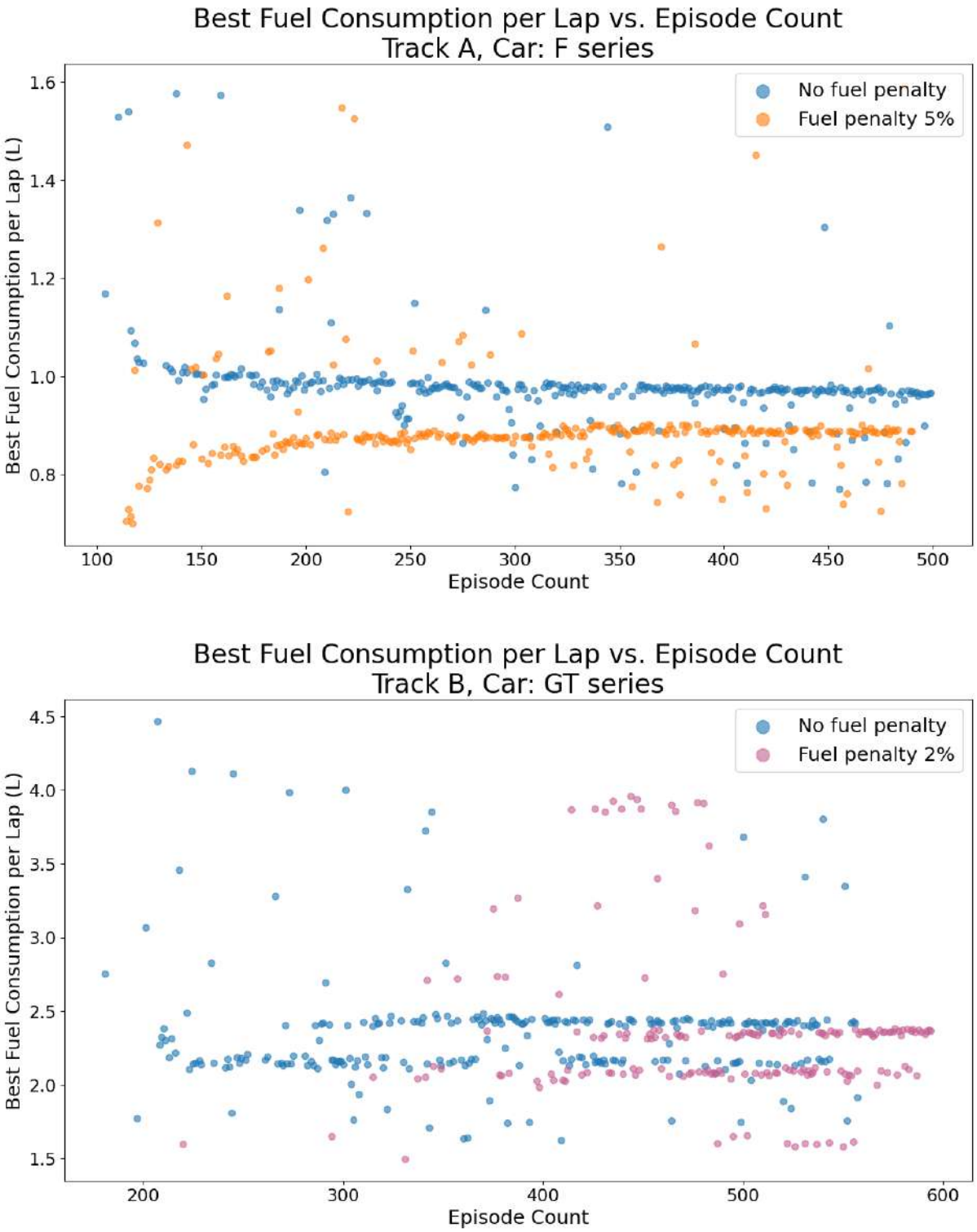


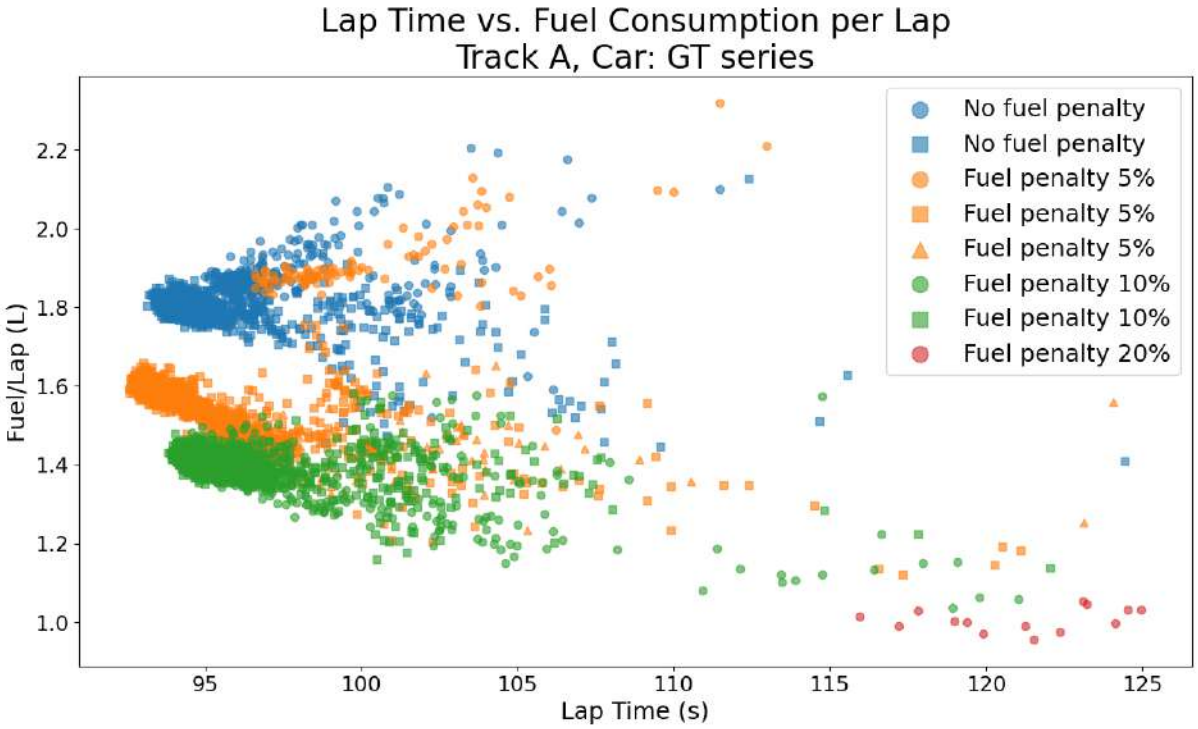
Figure 6. Best fuel consumption rate per lap across training episodes. Fuel consumption is measured in liters (L). Each subplot shows a specific car and track combination. Colors represent different fuel penalty levels. Shapes depict different random seeds. Fuel penalties consistently reduce consumption, especially on Track A.

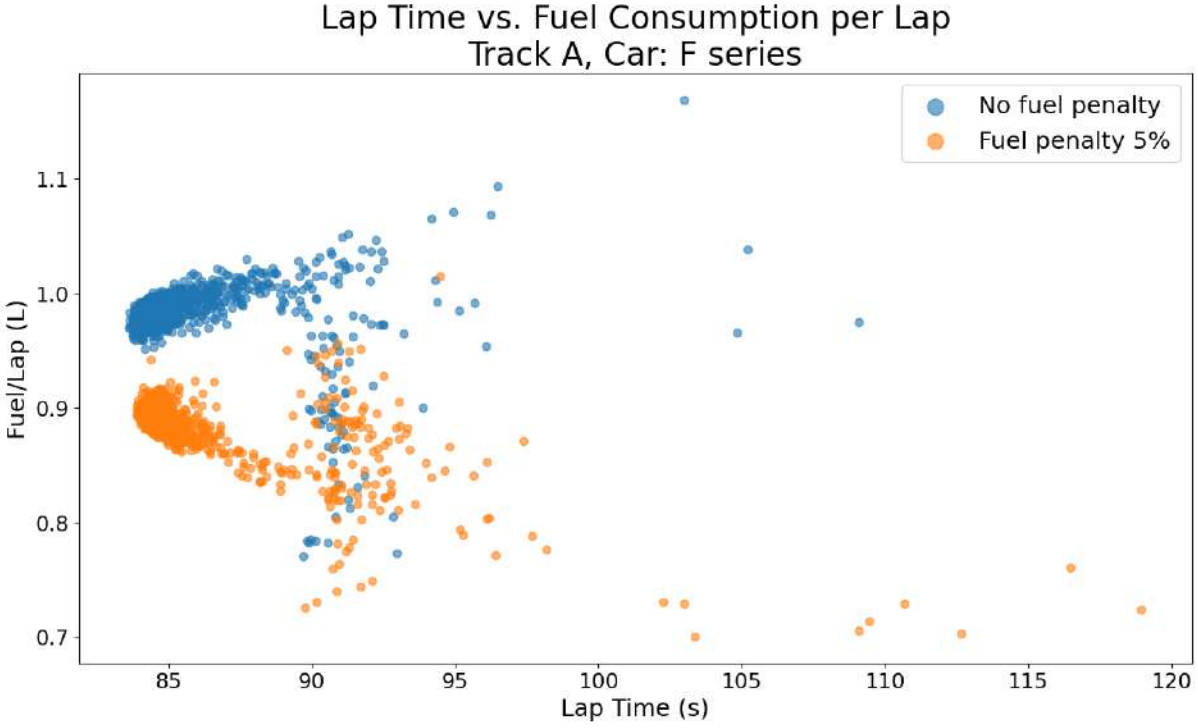
4.3 Trade-off Between Fuel Consumption and Lap Time

Figure 7 shows the relationship between lap time performance and fuel consumption per lap in different training scenarios. Overall, introducing a fuel penalty consistently led to more fuel-efficient driving strategies, often without significantly compromising lap time. For the 5% and 10% fuel penalty levels, this effect was most visible. The agents achieved significant fuel savings while maintaining lap times comparable to the baseline.

Another notable observation is the appearance of distinct clusters for each penalty level. This suggests that the agents converged to stable, consistent strategies, which differ for each reward configuration.

On Track A with the GT3 and F3 car, the agent maintained competitive lap times under 5% and 10%, penalties, while the 20% penalty resulted in substantially longer lap times and less effective performance. On Track B, two clusters also emerged for the 0% and 2% settings, though they were less distinct, likely due to the lower magnitude of the penalty.


Lap Time vs. Fuel Consumption per Lap
Track A, Car: GT series
No fuel penalty
No fuel penalty
Fuel penalty 5%
Fuel penalty 5%
Fuel penalty 5%
Fuel penalty 10%
Fuel penalty 10%
Fuel penalty 20%
Fuel/Lap (L)
Lap Time (s)


Lap Time vs. Fuel Consumption per Lap
Track A, Car: F series
No fuel penalty
Fuel penalty 5%
Fuel/Lap (L)
Lap Time (s)

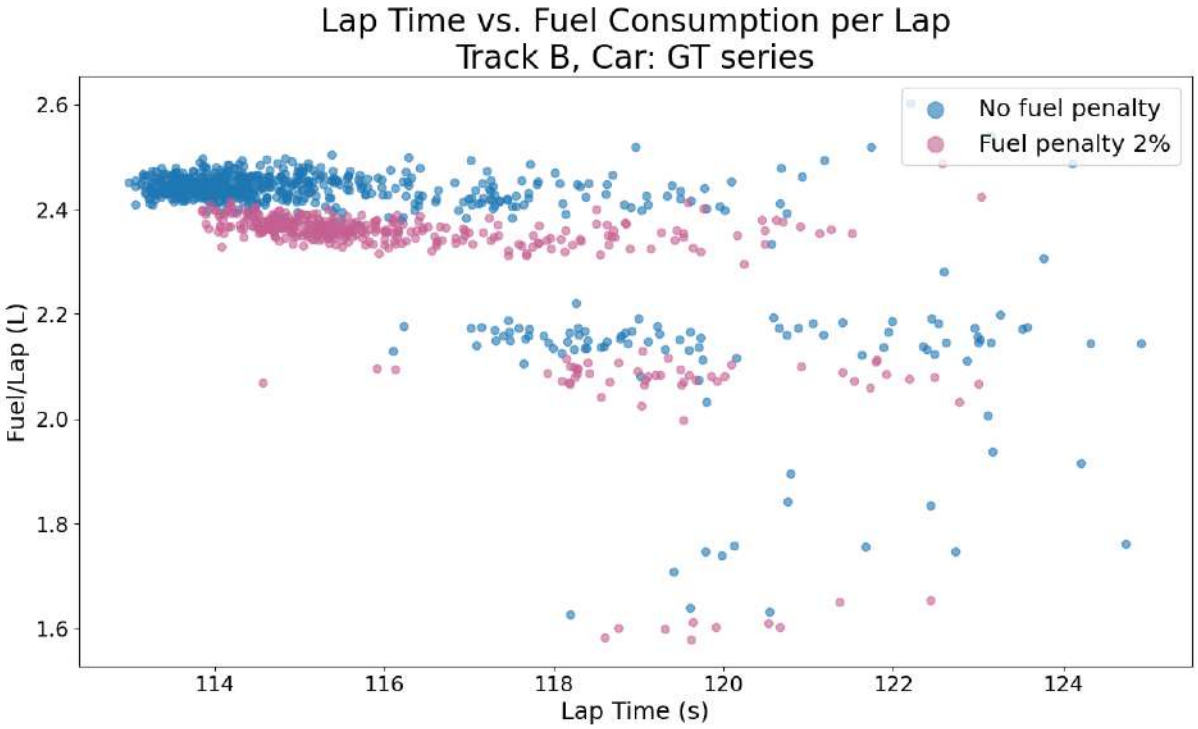


Figure 7. Scatter plots of lap time vs. fuel consumption per lap for different cars and tracks. Each point represents a completed lap. Colors indicate different fuel penalty levels, and marker shapes represent random seeds. Models trained with fuel penalties converged to more fuel-efficient driving behaviors, forming distinct performance clusters – especially on Track B.

To compare the performance of each model, we include a summary of lap time and fuel consumption statistics in Table 5. For every combination of car, track, and fuel penalty, we report the mean, standard deviation, minimum and maximum values for both lap time and fuel consumption per lap during training.

Track A, Car: GT3 series				
Fuel Penalty	**Count**	**Mean**	**Std**	**Max**
0%	1523	1.96	0.35	3.66
5%	1949	1.68	0.31	3.44
10%	2334	1.50	0.26	2.61
20%	26	1.13	0.23	1.59

Track A, Car: F3 series				
Fuel Penalty	**Count**	**Mean**	**Std**	**Max**
0%	1749	1.03	0.16	1.80
5%	1360	0.93	0.14	1.63

Track B, Car: GT3 series				
Fuel Penalty	**Count**	**Mean**	**Std**	**Max**
0%	836	2.56	0.48	4.47
2%	520	2.52	0.52	3.97

Table 5. Statistics of fuel consumption per lap during training across different fuel penalty levels, vehicles (GT3 and F3), and tracks (A and B). For each setup, the table reports the number of recorded laps (Count), mean fuel consumption, standard deviation, and maximum observed consumption. Results highlight the consistent reduction in average and maximum fuel usage when increasing penalty strength.

4.4 Testing comparison

To evaluate how Fuel Consumption (FC) varied based on driving behavior, we compared two models trained with different reward functions: no fuel penalty (0%) and a moderate fuel penalty (5%). Both models were trained using the GT3 car on Track A. To allow for fair comparison in terms of fuel efficiency, the models were selected based on having approximately the same lap time (92.964 seconds/lap and 92.784 seconds/lap, respectively). We provide a general overview of the comparison in Table 6.

Fuel Penalty	Best Lap Time (s) ↓	Mean Lap Time (s) ↓	FC for Best Lap (L) ↓	Mean FC Per Lap (L) ↓	Training Episodes ↓
0%	92.964	92.975	1.842	1.841	768
5%	**92.784**	**92.796**	**1.646**	**1.644**	**474**

Table 6. Comparison of two SAC models with different fuel penalties (0%, 5%) in the reward function during testing. The model trained with a 5% fuel penalty used approximately 10.7% less fuel per lap on average, model trained with 5% fuel penalty reduced its fuel usage by approximately 0.19 liters per lap (about 10.7%). and drove faster by 0.18 seconds (about 0.2%), while completing the training faster (trained for 38% less episodes)

4.5 Fuel Consumption Comparison

We first examined the remaining fuel over the lap distance travelled (Figure 8), to see how effective the fuel penalty was. As shown in the plot, the model trained with the 5% fuel penalty consistently consumed less throughout the duration of the lap, which resulted in total savings of approximately 0.2 liters per lap.

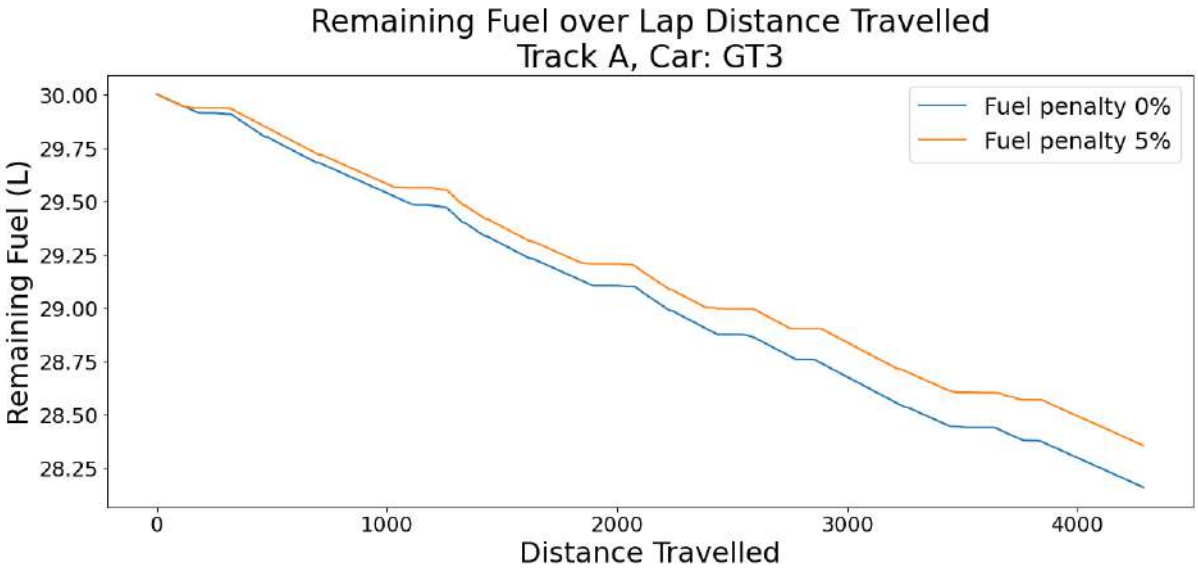


Figure 8. Remaining fuel over lap distance for two models trained with and without a fuel penalty (GT3 car, Track A). The model trained with a 5% fuel penalty consistently used less fuel throughout the lap, saving approximately 0.2 liters while maintaining a faster lap time.

To better understand where these fuel savings occurred, we visualized the track map with spatial differences in fuel consumption between the two models (Figure 9). We observed that the model trained with the fuel penalty consumed less fuel primarily in the sections leading into major turns. We hypothesized that these savings happen due to smoother acceleration profiles and gear management.

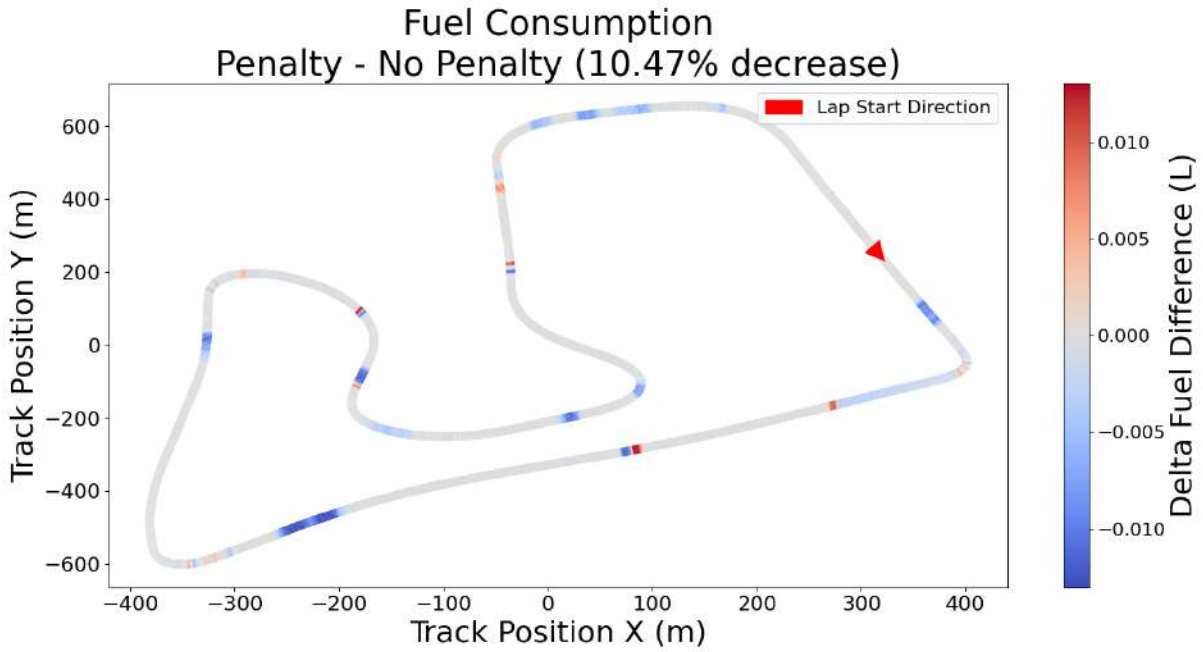


Figure 9. Spatial fuel consumption difference between the two models in Track A. Blue regions show where the 5% fuel-penalty model used less fuel, which usually happens right before major turns. Red regions show segments where the 5% fuel penalty model used more fuel. The fuel-penalty model reduced average fuel consumption for simulation step by 11.69% compared to the baseline.

4.6 Effects of Acceleration on Fuel Consumption

As was concluded in Table 2 in the earlier analysis, the acceleration status was the most important factor in the calculation of delta fuel. The fuel-efficient model learned to reduce fuel consumption before major turns by accelerating less aggressively in those sections. However, this is not a one-to-one correlation, as delta fuel also depends on other factors such as RPM and gear. Figure 10 shows a visual representation of acceleration status differences between two models. The graph looks very similar to the fuel efficiency difference graph due to the high impact of acceleration on fuel efficiency.

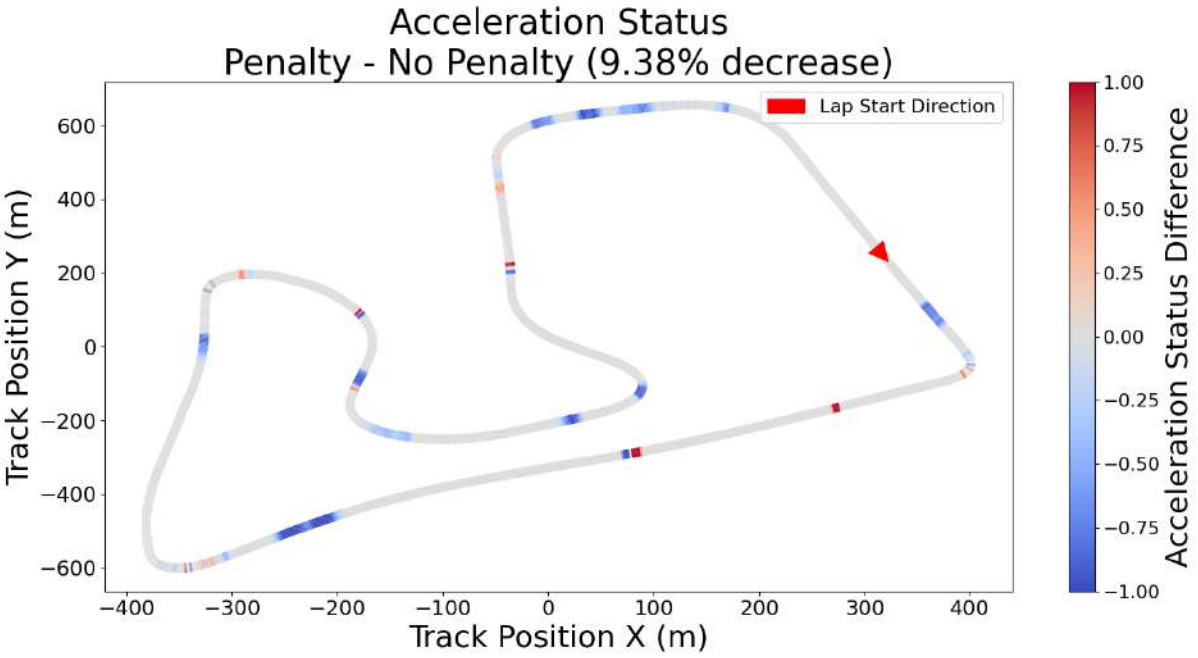


Figure 10. Map visualization of acceleration differences between the two models for Track A. The fuel-efficient model tended to accelerate less before turns (blue regions). The fuel-efficient model was able to reduce the average throttle input by 10.35% compared to the baseline.

To better understand the relationship in test cases, we plotted delta fuel against acceleration status (Figure 11) similarly to our analysis of training data in previous sections. While there is a general linear trend, the data shows significant variance. This is consistent with our earlier analysis of the fuel calculation and

suggests that while the acceleration input is the most significant factor in delta fuel, it is also important to include other factors engine RPM and driving behavior.

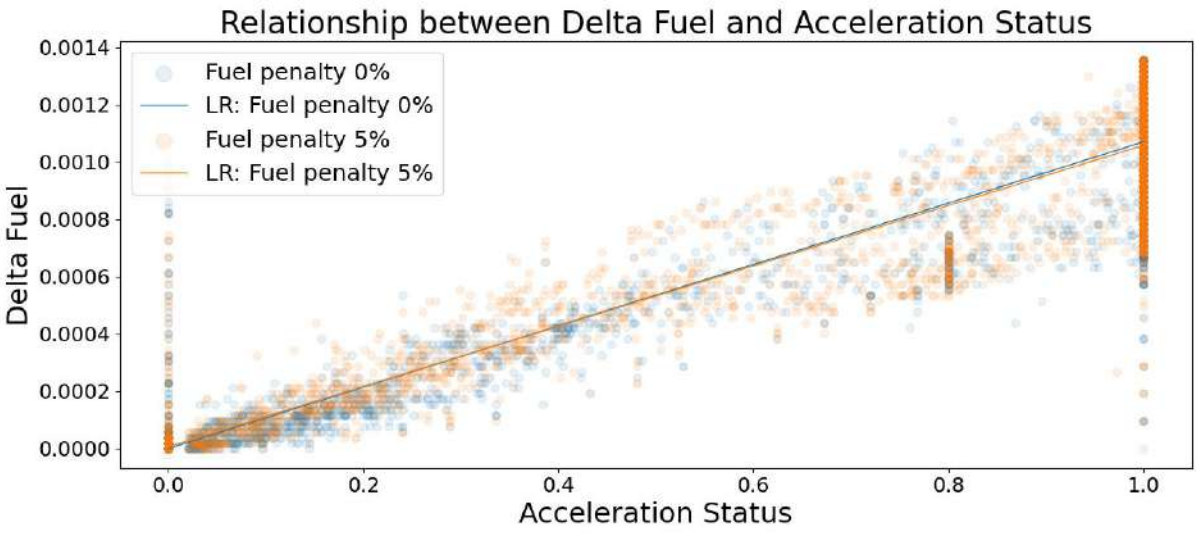


Figure 11. Scatter plot showing the relationship between acceleration status and delta fuel during testing for models trained with and without a fuel penalty. Each point represents a single time step from a completed lap. Linear regression (LR) lines are fitted for both configurations. Both models show a positive correlation between acceleration and fuel consumption.

4.7 Effects of Gear Usage and Engine RPM on Fuel Consumption

According to Pearson partial correlation coefficients from Table 2, engine RPM was the second most important factor affecting fuel consumption rates. We observed that the model trained with a fuel penalty learned to stay in higher gears, which helps it to lower engine RPM and reduce fuel consumption. Although the model had no direct control over the gearbox, it indirectly influenced gear shifts through maintaining a certain speed above the threshold to prevent downshifting. Figure 12 shows the gear selection over lap distance, and visualized on the track map, where the fuel-efficient model was able to avoid switching to fourth gear. The results are also visualized on a track map. To help improve the clarity of the

graphs, we have intentionally removed the transitions to first gear that occur briefly during gear shifts. This made the figures easier to read and less cluttered.

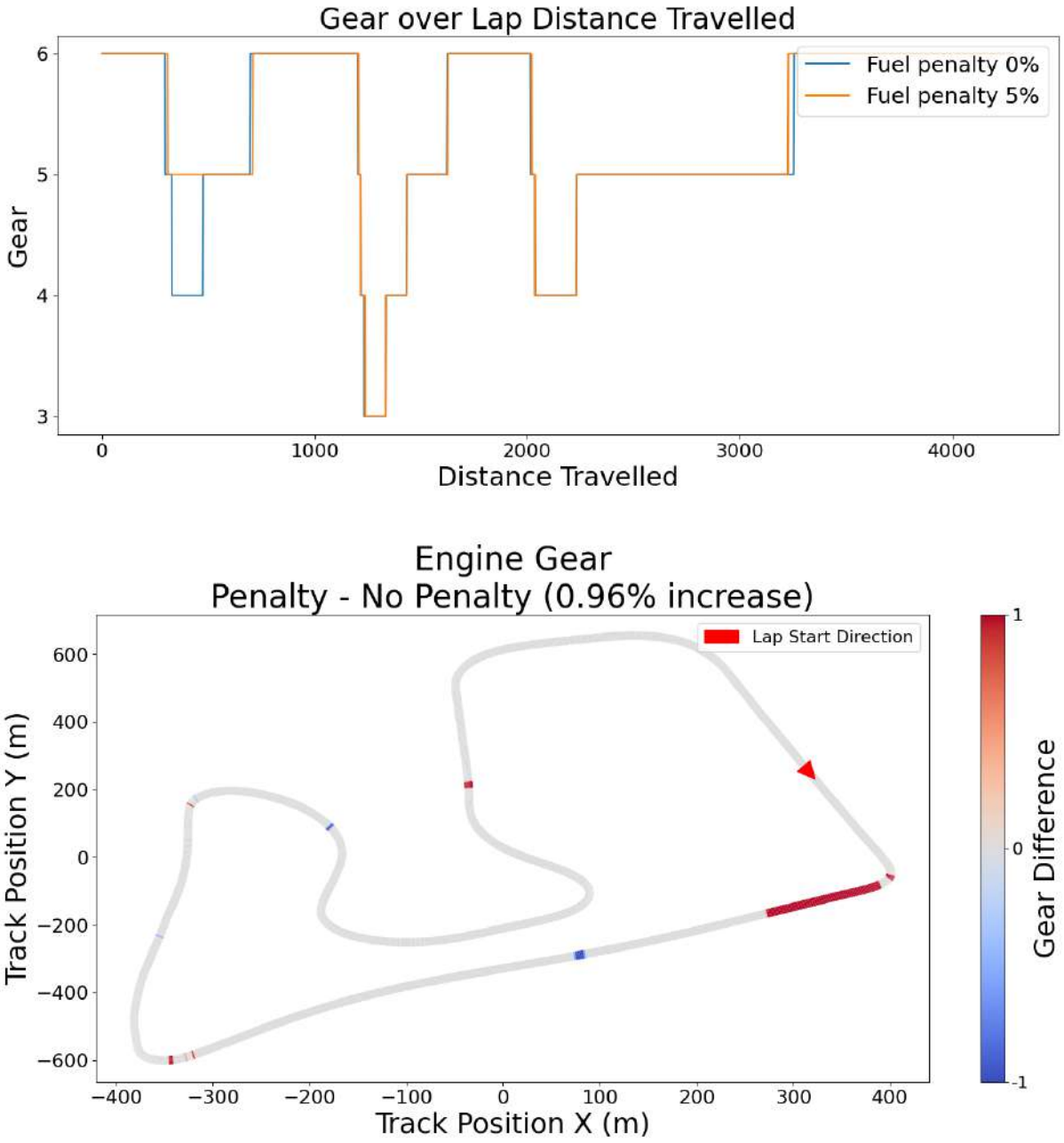


Figure 12. Gear usage comparison between the no-penalty and the 5% fuel penalty models. The top graph shows engine gear over the lap distance. The fuel-efficient model was able to maintain a higher gear in the beginning of the track. The bottom graph shows the gear differences visualized on a map of the track. Blue segments indicate sections on the track where the 5% fuel penalty model

remained in lower gears, while red segments show areas where it stayed in higher gears. It is evident that the engine avoided downshifting right after the first major turn of the track. The other short marks on the track indicate moments where the model downshifted or upshifted slightly earlier or later than the baseline. On average, the model was driving in higher gear for 0.96% longer.

To further explore how fuel consumption varied by gear for each model, we have calculated and plotted the distribution of delta fuel values across all possible gears in Table 7 and Figure 13. We observed that the model trained with a fuel penalty used less fuel across most gears, especially in 4th to 6th.

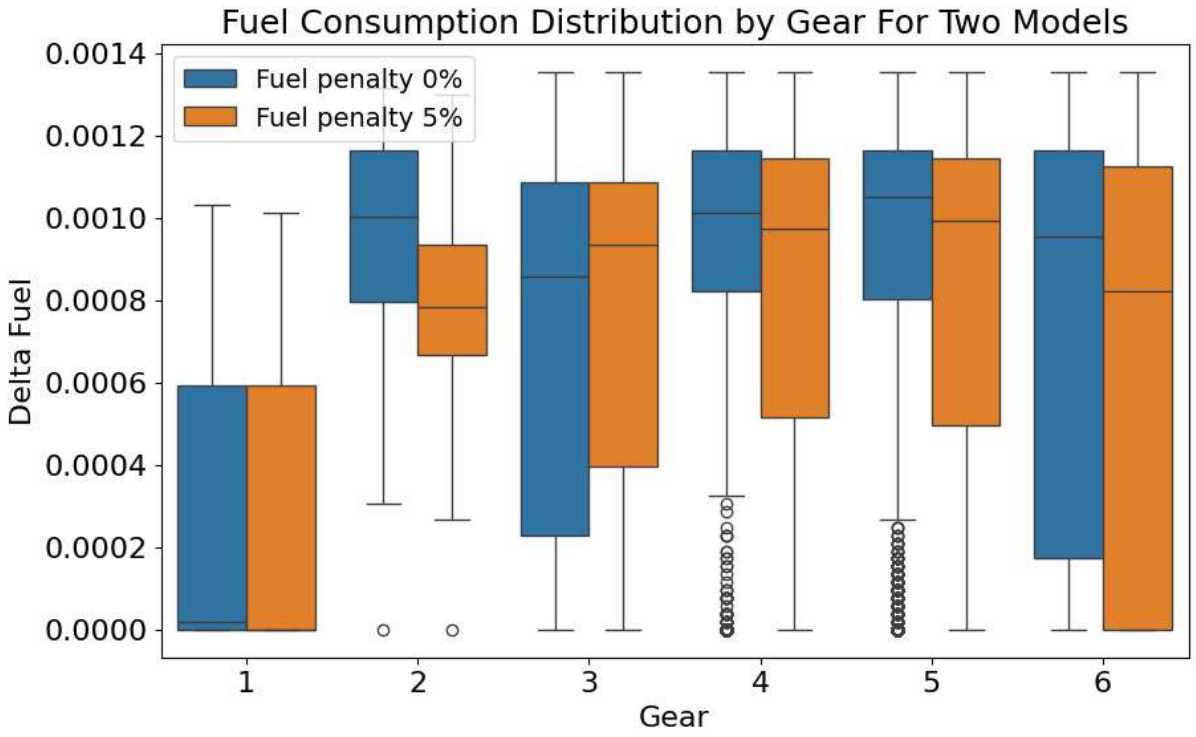


Figure 13. Distribution of delta fuel consumption per gear for models trained with and without a fuel penalty. The model trained with a 5% penalty (orange) showed consistently lower fuel usage in most gears, particularly in second gear and higher gears (4–6), where the car spent most of its time.

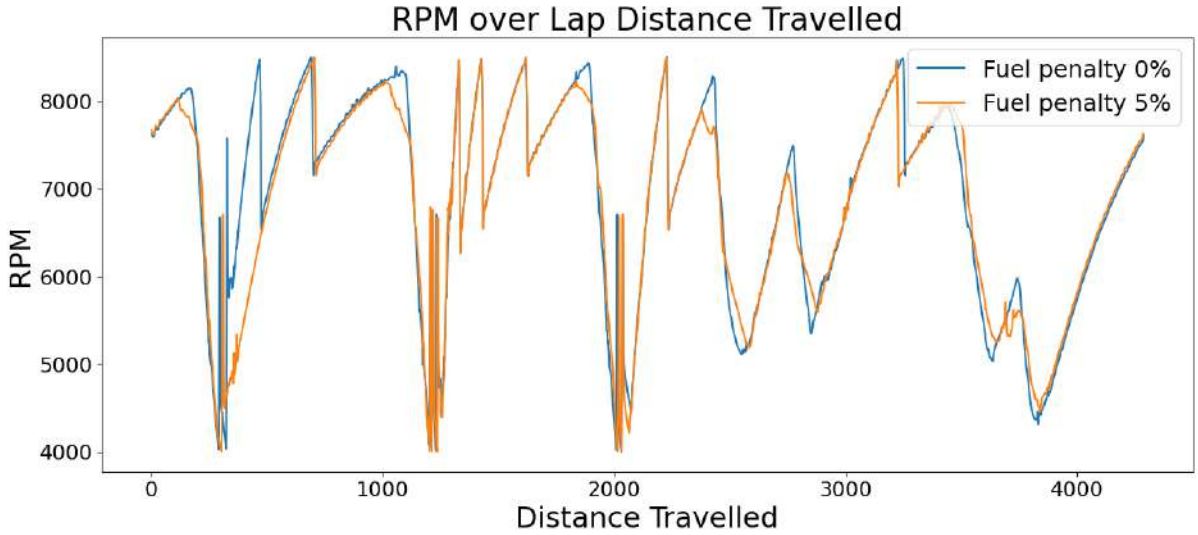
Gear	Fuel Penalty	Samples Count	Mean	Std	P-value
Gear 1	0%	224	2.73×10^{-4}	3.14×10^{-4}	0.68
	5%	204	2.60×10^{-4}	3.04×10^{-4}	
Gear 2	0%	84	9.59×10^{-4}	2.53×10^{-4}	0.00
	5%	88	7.87×10^{-4}	2.00×10^{-4}	
Gear 3	0%	451	7.17×10^{-4}	4.40×10^{-4}	0.69
	5%	424	7.40×10^{-4}	4.45×10^{-4}	
Gear 4	0%	1430	8.78×10^{-4}	4.09×10^{-4}	0.00
	5%	1189	7.96×10^{-4}	4.54×10^{-4}	
Gear 5	0%	3274	8.51×10^{-4}	4.46×10^{-4}	0.00
	5%	3564	7.96×10^{-4}	4.60×10^{-4}	
Gear 6	0%	4284	7.68×10^{-4}	4.82×10^{-4}	0.00
	5%	4266	6.49×10^{-4}	5.06×10^{-4}	

Table 7. Fuel consumption statistics by gear for models trained with and without fuel penalty across the entire testing session. The table reports the number of simulation steps spent in each gear (n_samples), the mean fuel consumption (delta_fuel), and the standard deviation of that distribution. The final column shows the p-value from a Mann-Whitney U test, which estimates the likelihood that the two samples come from the same distribution. Values below 0.05 for p-value generally mean a significant difference between samples. This is the case for gear 2 and gears 4 through 6. However, gear 2 is less reliable due to the small number of samples. Overall, the results indicate that the fuel-penalty model learned to manage its fuel consumption in higher gears more effectively.

As mentioned, engine RPM was the second most important factor that contributed to fuel consumption after the acceleration status. To further how the model

adapted the RPM of the car engine, we compared the RPM values between the no-penalty and the 5% penalty models. Overall, the 5% penalty model reduced its average engine gear by 0.88%.

Similarly to previous components, we visualized RPM differences between the two models across the track map in Figure 14 to better understand this relationship. We have concluded that RPM differences occurred in sections following major turns, which aligns with gear usage statistics.



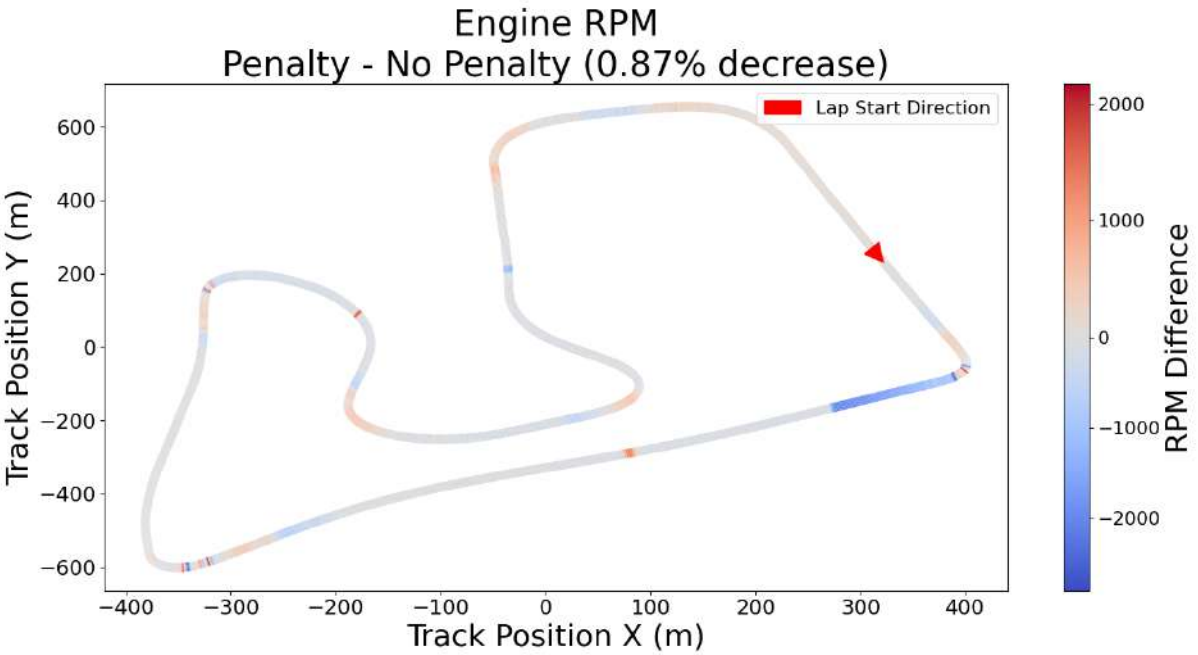


Figure 14. The top graph shows engine RPM over lap distance for two models with and without a fuel penalty (GT3 car, Track A). The model trained with a 5% fuel penalty (orange) consistently maintained lower RPM across the lap. The bottom graph shows the track map showing RPM differences between the two models across the lap. Blue segments indicate areas where the model with the 5% fuel penalty operated at a lower RPM, and red segments indicate a higher RPM. The most significant difference occurred right after the first major turn of the track, where the fuel-penalty model maintained higher gear. In addition, the fuel-penalty model maintained a higher RPM in the sections right before and even during the turns.

Additionally, we plotted delta fuel against RPM values for both models during testing in Figure 15. This allowed us to evaluate how much lowering RPM reduces fuel consumption and how this relationship was different for models trained with and without fuel penalty.

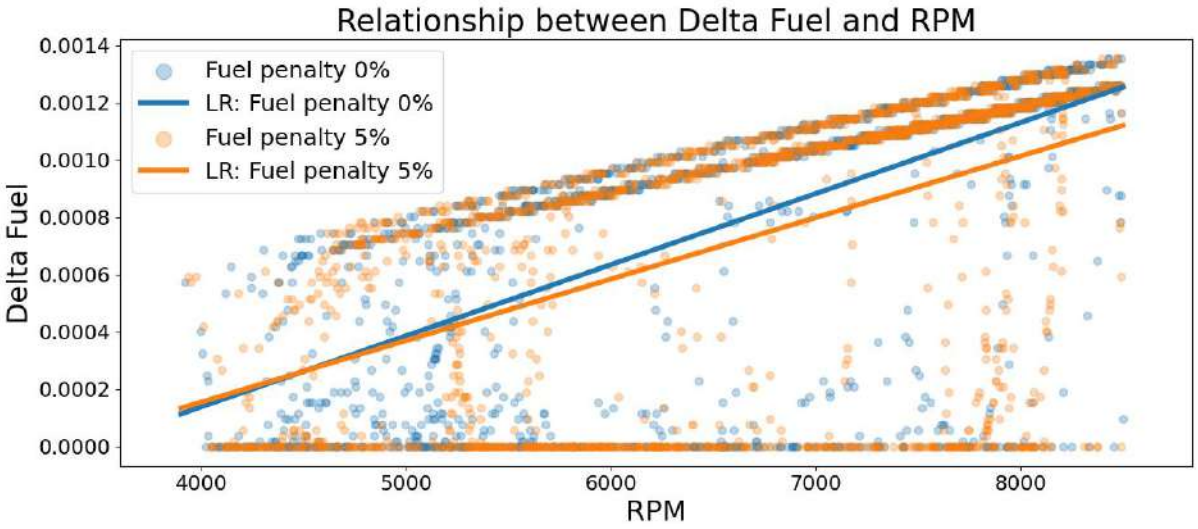
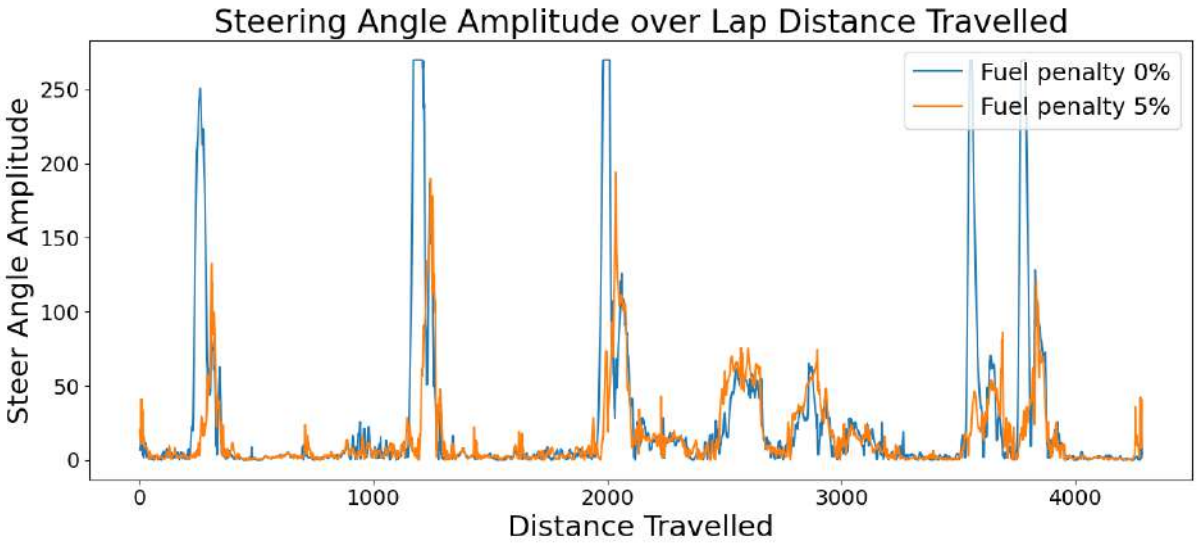


Figure 15. Scatter plot showing the relationship between engine RPM and delta fuel during testing. The graph displays data from both models and linear regression (LR) lines. Although delta fuel increases with RPM for both models, the 5% penalty model consistently consumes less fuel at similar RPM levels.

4.8 Effects of Steering Angle Amplitude on Fuel Efficiency

According to Table 2 from our previous analysis, we confirmed that the magnitude of the steering input (steering angle amplitude) was another factor that affected fuel consumption in the simulation. We hypothesized that this was because frequent oversteering could lead to higher rolling resistance, tire scrubbing, and more directional changes, which results in energy losses. Additionally, we believe that the steering behavior serves as a proxy for driver efficiency, which is why we deemed it important to analyze the difference in this component between the two models.

We observed that the model trained with a fuel penalty was able to reduce its average steering amplitude by 32.91% compared to the baseline. To better understand where these differences occur on the track, we visualized the spatial difference in steering angle amplitude between the two models (Figure 16).



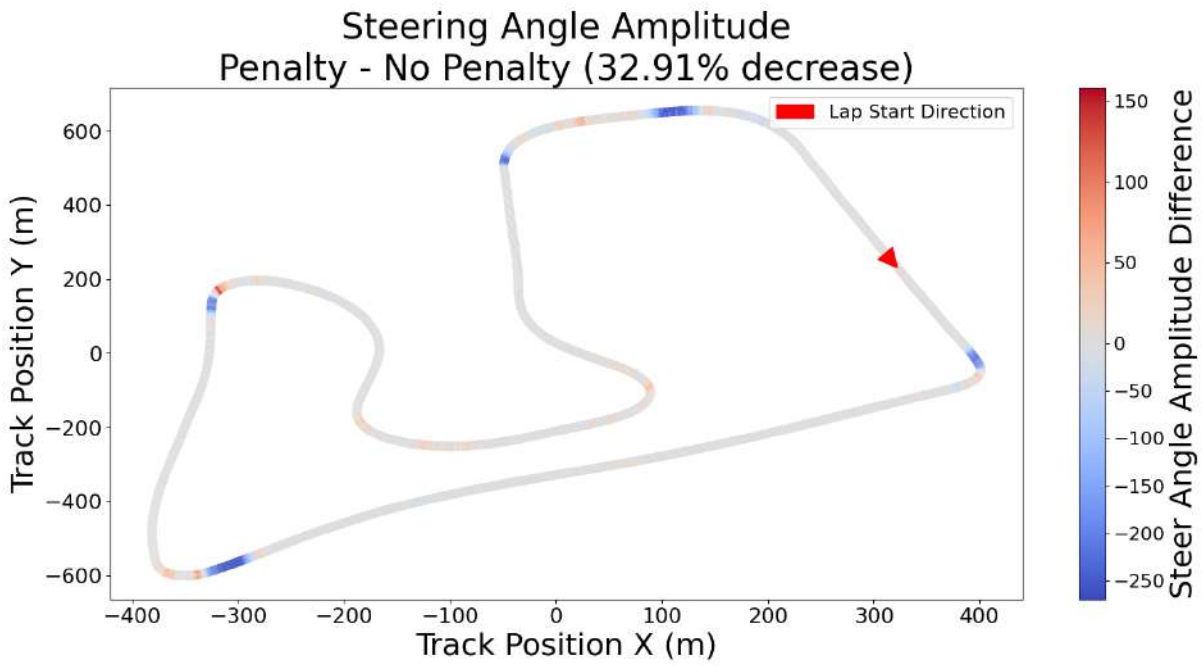


Figure 16. The top graph shows steering angle amplitude over lap distance for two models. The model trained with a 5% fuel penalty (orange) consistently showed a smaller steering angle than the baseline model (blue). The bottom graph visualizes the difference on the track map. Blue regions indicate regions where the fuel-penalty model reduced its steering inputs compared to the baseline model, and red regions show higher amplitudes for the fuel-penalty model. Most noticeable differences occur right before high-speed corners.

However, it is worth noting that according to our previous analysis, while steering angle amplitude moderately affects fuel use, it is not the most important factor. And therefore, when we plotted delta fuel against the raw steering angle amplitude for both models and fitted the linear regression lines in Figure 17, we did not observe a direct linear relationship. This suggests that the relationship between these factors is comparatively weaker and gets overshadowed by more direct factors like acceleration status and engine RPM.

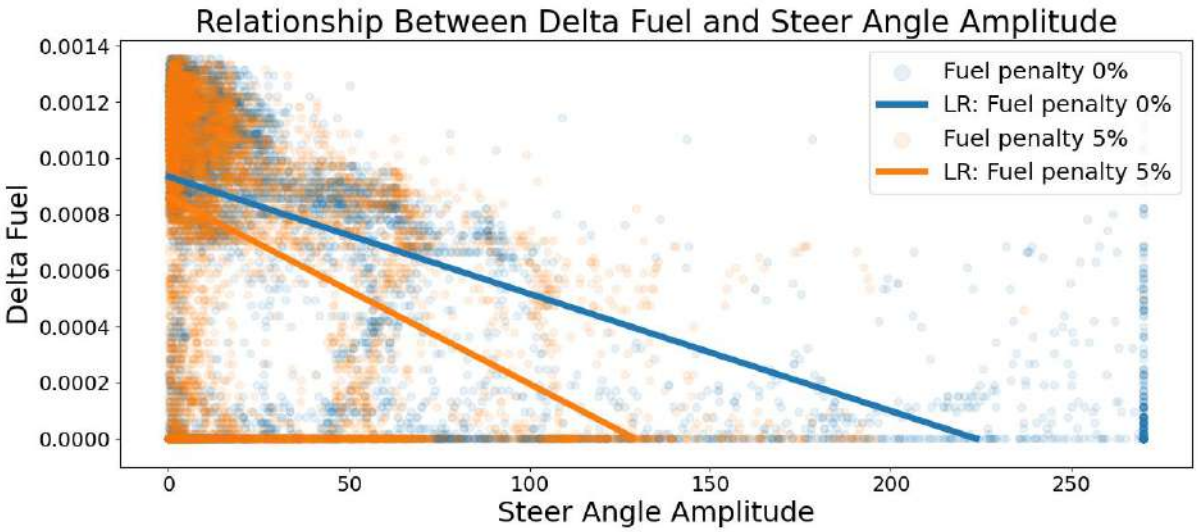


Figure 17. Scatter plot of delta fuel versus steering angle during testing. Data points and regression lines are shown for both models. The lack of a clear trend indicates that fuel consumption does not strongly correlate with steering input. This suggests that steering behavior insignificantly affects fuel consumption compared to other factors, or it does so indirectly.

To conclude, we observed that the model trained with a fuel penalty has learned to reduce its steering angle amplitude, even with a comparatively moderate correlation between the steering angle and the fuel consumption rate according to Table 2, compared to metrics such as acceleration status and engine RPM.

4.9 Lap Time Comparison

Lastly, it was established that the model with fuel penalty was able to complete its best lap in 92.78 seconds, compared to the no-penalty model that did it in 92.964. Thus, the model with the fuel penalty was able to save approximately 0.184 seconds, an improvement of almost 0.2%.

We wanted to explore in what sections of the track the time savings were achieved. Our findings in Figure 18 show that the model was able to save time in the sections right after major turns but was struggling in tight technical sections

compared to the baseline model. We hypothesized that due to smoother acceleration and braking profiles, the model intentionally learned to lose less speed going into the turn, which can lead to suboptimal driving lines and make it more difficult to recover in technical sections when turns happen in rapid succession. This aligns with the observation that the model was able to maintain higher gear when coming out of the first turn since it kept more speed. Additionally, the model did not accelerate as aggressively in the straight line and started to slowly lose its time advantage, likely prioritizing a less aggressive strategy when approaching the second turn.

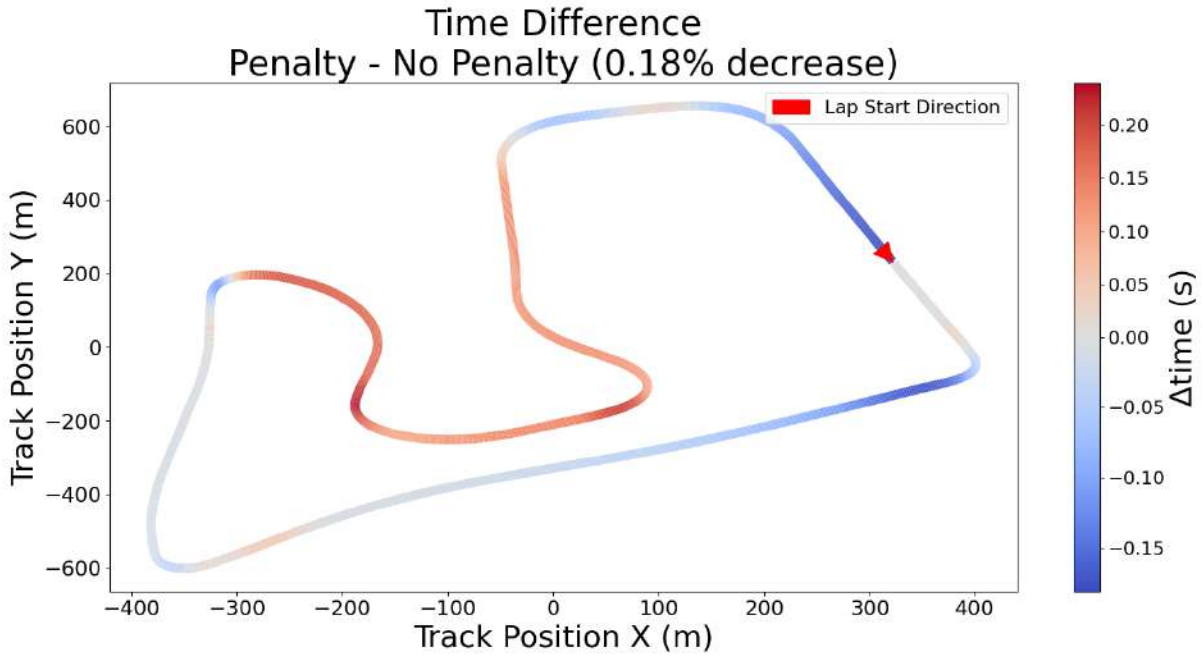


Figure 18. Time difference between the fuel penalty model and the model without the penalty across the track. Red regions show segments where the fuel penalty model was slower, and blue regions indicate where it was faster. The modified model gained time in sections after major turns. However, it loses in more technical segments and on high-speed straights.

5 RESULTS

Based on our experiments, we concluded that introducing the fuel consumption penalty into the reward function contributes to more fuel-efficient learning policies without substantially compromising lap time performance. In several configurations, especially on easier tracks, agents trained with mild fuel penalties not only matched but outperformed the baseline in both lap time and fuel efficiency.

5.1 Learning Performance and Lap Times

We observed that low to moderate fuel penalties (2%-5%) often resulted in faster convergence early in training on easier tracks, particularly on Track A. When the fuel penalty was set too high (10%-20%), the agent's performance started to decrease, and it started to prioritize fuel savings at the cost of lap time. However, this trend did not generalize to a more difficult Track B, when the model performance was already significantly affected at 2% fuel penalty, and models with 5% fuel penalty and more failed to complete a single lap.

This suggests that the optimal balance between fuel efficiency and performance varies for each track. On easier tracks like Track A, there is more room for fuel-saving maneuvers that work without affecting lap time. However, on difficult technical circuits like Track B, introducing even a mild fuel penalty can create an environment too difficult for the agent to learn effectively. For this reason, it is important to tune reward coefficients based on specific configurations.

5.2 Fuel Consumption Trends

Introducing a fuel penalty consistently reduced fuel consumption across all scenarios. On Track A, both models trained at the 5% and 10% penalty levels demonstrated significant savings without compromising lap time. On average, the model trained with a 5% fuel penalty was able to save about 0.19 liters of fuel (approximately a 10.7% decrease) while completing the lap faster by 0.18 seconds (almost a 0.2% improvement).

During the experiments, we noticed the appearance of distinct fuel consumption per lap clusters associated with each penalty level (Figure 7). These clusters represent the most efficient strategies that the model developed for each fuel penalty level. Depending on the strength of the penalty, the optimal policy changes, and that is reflected by different clusters in the graph.

5.3 Trade-offs and New Strategies

The agent learned key driving strategies to reduce fuel usage that mimic real-world energy-saving strategies used in motorsport, such as maintaining momentum and staying in higher gears. Interestingly, while human drivers often use early braking as a fuel-saving strategy, the agent did not replicate this behavior. We hypothesize that since early braking tends to increase overall lap time, the agent trained with a lower fuel penalty (5%) chose not to adopt this tactic to maintain a competitive lap time.

5.4 Component-Level Analysis

We analyzed the features present in the simulator state and used the partial correlation coefficients to select three main components that affect immediate fuel usage the most: acceleration status, engine RPM, and steering angle amplitude (Table 2).

Further detailed analysis of these features confirmed that the model adapted its driving style to allow for more fuel savings. In particular, the model started using less aggressive acceleration strategies, particularly before sharp turns and managed to lower engine RPM by keeping the higher gear after tight corners. Additionally, although steering amplitude had only a moderate correlation with fuel use, the agent trained with the fuel penalty was able to reduce and stabilize its steering amplitude, more closely mimicking human driving.

These results show that the SAC algorithm was able to adapt not only the simplest fuel-efficient behaviors but also secondary effects that improved the agent's stability.

6 DISCUSSIONS

Our work highlights the ability of reinforcement learning agents to adopt fuel-efficient behaviors given appropriate reward shaping. However, several limitations remain that could be explored in future work.

First, all experiments were conducted using only two vehicle models and two racetracks. The current setup raises concerns about the generalizability of the algorithm. Especially considering that the agent's performance from simpler tracks such as Track A did not transfer to more technical tracks like Track B.

Another limitation is the lack of experiments with manual gear-shifting mechanisms. During the experiments, the agent had no direct control of the gearbox and could only influence it indirectly by adjusting the speed of the car. This prevented the model from using realistic driving techniques often used to conserve fuel, such as short-shifting.

A third constraint is that the Assetto Corsa simulator only operates in real-time, which considerably slows down training. As mentioned, we needed roughly 48 hours to complete 500 training episodes. This limits our ability to test alternative reward function designs or repeat training using different random seeds to ensure stability.

Additionally, only a single reward function design was explored. Although we tested this design with different penalty levels, exploring alternative reward functions would provide more insight into the ability of SAC to manage resource constraints. Future work could include a short history of past fuel usage into the state space or directly penalize underlying factors that influence fuel use rather than fuel usage alone.

Moreover, the fuel usage metric was excluded from the observation space of the agent to simulate partial observability. While it did not significantly affect the ability of the agent to develop new strategies, it might be worth exploring setups that include this information in the observation space.

7 CONCLUSIONS

Our work explored the ability of reinforcement learning agents in autonomous racing environments to adapt to multi-objective tasks that involve optimizing both lap times and energy efficiency. We incorporated a fuel usage penalty into the reward function and demonstrated that agents trained with low to moderate penalties can achieve considerable fuel savings without significantly compromising their lap-time performance. The agents adapted by modifying their driving behaviors, such as reducing acceleration, managing engine RPM through gear changes, and even increasing steering smoothness. However, these effects

did not transfer to more complex tracks, where even low penalties severely affected the agents' abilities to learn. This highlights the need for more careful penalty calibration depending on the parameters of the environment.

Overall, our results suggest that reinforcement learning can be effectively used to balance performance and energy efficiency. Future work could focus on generalizing these strategies across additional tracks and vehicles and could explore alternative reward function designs or observational conditions.

8 ACKNOWLEDGMENTS

The Assetto Corsa simulation environment was used solely for the purposes of this research. We do not intend to use it in any public-facing or commercial context, nor in any activity involving public distribution or display.

The simulator is publicly available on the Steam platform. However, for using Assetto Corsa for research purposes, it is required to obtain the appropriate permissions from the Assetto Corsa Support Team.

9 REFERENCES

[1] Hu et al., "How Simulation Helps Autonomous Driving: A Survey of Sim2real, Digital Twins, and Parallel Intelligence", doi:10.1109/TIV.2023.3312777, Jan. 2024.

[2] Werner et al., “Accelerating Autonomy: Insights from Pro Racers in the Era of Autonomous Racing - An Expert Interview Study”, arXiv:2405.02620, May 2024.

[3] Jung et al., “An Autonomous System for Head-to-Head Race: Design, Implementation and Analysis; Team KAIST at the Indy Autonomous Challenge”, arXiv:2303.09463, Mar. 2023.

[4] Remonda et al., “A Simulation Benchmark for Autonomous Racing with Large-Scale Human Data”, Jul. 2024.

[5] Evans et al., “Unifying F1TENTH Autonomous Racing Survey, Methods and Benchmarks”, arXiv:2402.18558, Feb. 2024.

[6] Remonda et al., “Formula RL: Deep Reinforcement Learning for Autonomous Racing using Telemetry Data”, arXiv:2104.11106, Apr. 2021.

[7] Fuchs et al., “Super-Human Performance in Gran Turismo Sport Using Deep Reinforcement Learning”, arXiv:2303.09463, Mar. 2023.

[8] Vasco et al., “A Super-human Vision-based Reinforcement Learning Agent for Autonomous Racing in Gran Turismo”, arXiv:2406.12563, Jun. 2024.

[9] Tammewar et al., “Improving the Performance of Autonomous Driving through Deep Reinforcement Learning”, doi:10.3390/su151813799, Aug. 2023.

[10] Kiran et al., “Deep Reinforcement Learning for Autonomous Driving: A Survey”, arXiv:2002.00444, Feb. 2020.

[11] Ashraf et al., “Optimizing hyperparameters of deep reinforcement learning for autonomous driving based on whale optimization algorithm”, doi: 10.1371/journal.pone.0252754, Jun. 2021.

[12] S. Aradi, “Survey of Deep Reinforcement Learning for Motion Planning of Autonomous Vehicles”, arXiv:2001.11231, Jan. 2020.

[13] Tang et al., “Highway Decision-Making and Motion Planning for Autonomous Driving via Soft Actor-Critic”, doi:10.1109/TVT.2022.3151651, May 2022.

[14] Niu et al., “Two-Stage Safe Reinforcement Learning for High-Speed Autonomous Racing”, doi:10.1109/SMC42975.2020.9283053, Oct. 2020.

[15] Song et al., “Autonomous Overtaking in Gran Turismo Sport Using Curriculum Reinforcement Learning”, arXiv:2103.14666, Mar. 2021.

[16] Huang et al., “End-to-End Autonomous Driving Decision Based on Deep Reinforcement Learning”, doi:10.1109/ICCAR.2019.8813431, Apr. 2019.

[17] Elallid et al., “A Comprehensive Survey on the Application of Deep and Reinforcement Learning Approaches in Autonomous Driving”, doi:10.1016/j.jksuci.2022.03.013, Oct. 2022.

[18] Kim et al., “Autonomous Vehicle Fuel Economy Optimization with Deep Reinforcement Learning”, doi:10.3390/electronics9111911, Oct. 2020.

[19] Betz et al., “Autonomous Vehicles on the Edge: A Survey on Autonomous Vehicle Racing”, arXiv:2202.07008, Feb. 2022.

[20] Revell et al., “Sim2real: Issues in transferring autonomous driving model from simulation to real world”, doi:10.1109/SoutheastCon48659.2022.9764110, Mar. 2022.

[21] Han et al., “A Survey on Deep Reinforcement Learning Algorithms for Robotic Manipulation”, doi:10.3390/s23073762, Mar. 2023.

[22] Nguyen et al., “Review of Deep Reinforcement Learning for Robot Manipulation”, doi:10.1109/IRC.2019.00120, Feb. 2019.

[23] Mnih et al., “Playing Atari with Deep Reinforcement Learning”, arXiv:1312.5602, Dec. 2023.

[24] Schulman et al., “Proximal Policy Optimization Algorithms”, arXiv:1707.06347, Jul. 2017.

[25] Haarnoja et al., “Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor”, arXiv:1801.01290, Jan. 2018.

[26] R. Sutton & A. Barto, “Reinforcement Learning: An Introduction”, ISBN: 9780262193986, Feb. 1998.

[27] Wurman et al., “Outracing champion Gran Turismo drivers with deep reinforcement learning”, doi:10.1038/s41586-021-04357-7, Aug. 2021.

[28] Brockman et al., “OpenAI Gym”, arXiv:1606.01540, Jun. 2016.

[29] Dosovitskiy et al., “CARLA: An Open Urban Driving Simulator”, arXiv:1711.03938, Nov. 2017.

[30] Krajzewicz et al., “SUMO (Simulation of Urban MObility) - an open-source traffic simulation”, ISBN:90-77039-09-0, Jan. 2002.

[31] Wymann et al., “Torcs, the open racing car simulator”, Software available at http://torcs.sourceforge.net, Jul. 2000.

[32] Tian et al., “Balanced Reward-inspired Reinforcement Learning for Autonomous Vehicle Racing”, 6th Annual Learning for Dynamics & Control Conference (pp. 628-640), Jun. 2024.

[33] S. Alighanbari & N. Azad, “Deep Reinforcement Learning With NMPC Assistance Nash Switching for Urban Autonomous Driving”, doi:10.1109/TIV.2022.3167616, Apr. 2022.

[34] Herman et al., “Learn-to-Race: A Multimodal Control Environment for Autonomous Racing”, arXiv:2103.11575, Mar. 2021.

[35] D. Li & O. Okhrin, “A Platform-Agnostic Deep Reinforcement Learning Framework for Effective Sim2Real Transfer towards Autonomous Driving”, arXiv:2304.08235, Apr. 2023.

[36] “The Future of Transportation – Salvomie Electrical innovations.” Available at https://www.salvomie.co.uk/the-future-of-transportation/, Dec. 2023.