Full-scale Autonomous Driving: Simulation-Based Policy Comparison

Sean Li & Jimmy Sun

11 November 2023

Contents

Abstract

This report contains a simulation for an autonomous driving car aimed at evaluating decision-making policies for a self-driving car on a high-speed and complicated road. Using the Autonomous Driving Gymnasium framework, we compared a range of different approaches including Reinforcement Learning to determine the optimal implementation.

We developed a range of road conditions, defined key performance metrics, and executed a large number of tests to measure each policy's effectiveness. The training was done on a Linux PC 24 hours a day, 7 days a week with powerful hardware including Intel Core i9 13900KS and Nvidia RTX 4090. Multiple algorithms were spread across the CPU and GPU for efficient use of resources.

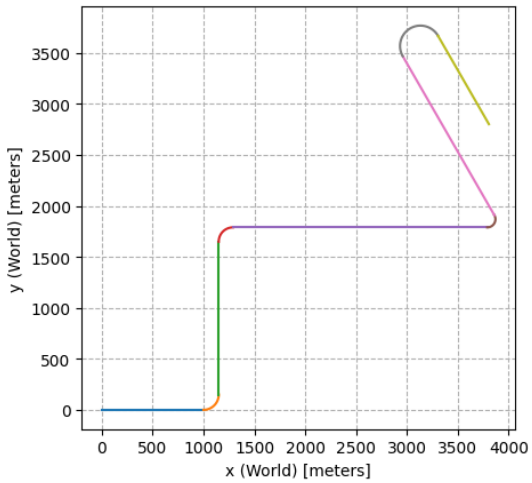
The findings reveal significant insights into the policies under varied conditions. The report includes a discussion on the practicality of the strategies based on simulation outcomes, alongside considerations from industry perspectives gained through guest lectures.

1 Creating Simulation Roads

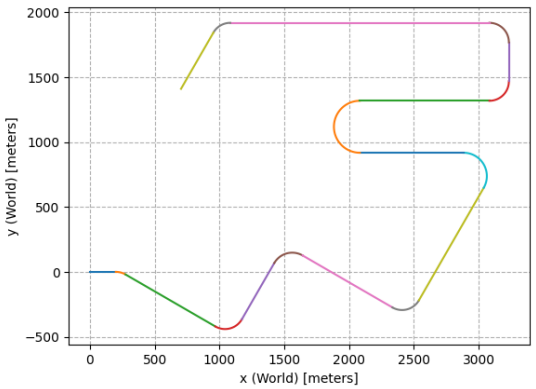
The primary objective was to design simulation roads that replicate real-world driving conditions, particularly focusing on high-curvature turns and comprehensive driving scenarios. The roads used for training and evaluation were carefully chosen, keeping in mind the challenges of autonomous driving under various challenging circumstances.

1.1 Design of Simulated Roads

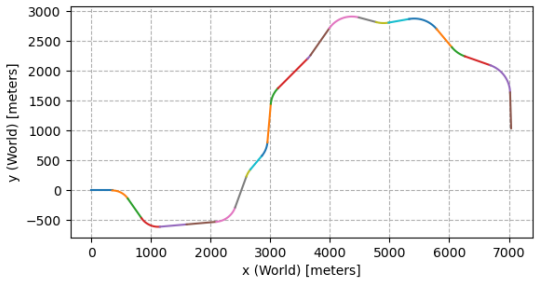
We have designed five distinct roads, each characterised by unique elements and challenges, as detailed in Table 1. The designs incorporate sharp turns, 180-degree curves, and a combined length approaching 10km to simulate extensive driving conditions.



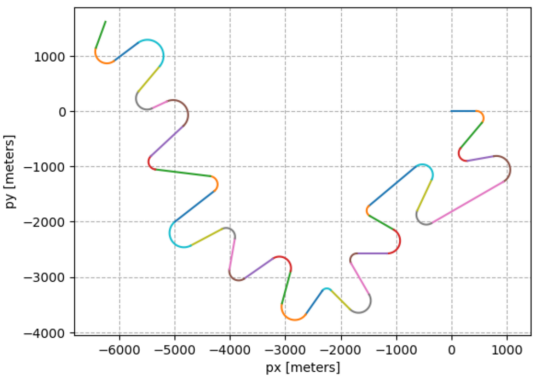
(a) Testing Road A



(b) Testing Road B



(c) Training Road C



(d) Training Road D

Figure 1: Road Designed for RL training & testing

1.2 Rationale Behind the Design

The design choices were based on constructing roads that simulates the unpredictability of real-world driving. Testing Roads A (Figure 1a) and B (Figure 1b) were designed to to have sharp turns with curvatures as high as 1/50, and angles as much as 120 degrees,

Road Name	Figure	Length (meters)
Testing Road A	Figure 1a	9051
Testing Road B	Figure 1b	10334
Training Road C	Figure 1c	10536
Training Road D	Figure 1d	24785

Table 1: Names and lengths of the simulated roads

as well as 180 degree U-turns. These roads, with their high curvature and continuous turns, require careful maneuvering in decision-making algorithms.

However, recognising the constraints on training resources and time, even though Intel i9 and RTX 4090 were used, the training still took a considerable time. We optimised the design for Training Roads C (Figure 1c). While shorter, these roads encapsulate all the challenges present in real world roads, thereby maintaining the comprehensiveness of the training phase without the excessive computational demand. This approach significantly reduced the average training time from more than 10 hours to a more manageable a few hours. In addition, once we determine the policy works on a smaller road, we perform another training on a longer and much more complicated road, as shown in Figure 1d. This road is 25 km in total and has continuous sharp corners, U-turns and straight sections.

2 Defining Performance Metrics

2.1 Progress Reward

The "progress reward" function motivates the autonomous vehicle to advance effectively along its intended route. By assessing the progress made from one time step to the next, this function rewards forward movement, directly aligning with the vehicle's primary objective: reaching its destination. This reward is fundamental in ensuring that the vehicle's actions contribute positively to journey progression, avoiding unnecessary detours or actions that might prevent timely arrival at the destination. This is particularly important for the vehicle to drive forward, rather than staying in one location to obtain maximum reward.

2.2 Distance Reward

The "distance reward" function is a critical component in autonomous vehicle reinforcement learning, encouraging the vehicle to maintain an optimal and reasonable position relative to the centre of the road. By calculating rewards based on the distance from the road's centre, the system incentives actions that align with safe, centred driving behaviours. Moreover, the training and simulation episode will be terminated if the vehicle deviates too far from the centre.

2.3 Speed Reward

The "speed reward" function plays a pivotal role in regulating the autonomous vehicle's velocity, ensuring it stays within safe and legal limits while optimising travel time.

This function generates rewards by assessing the car's current speed, encouraging acceleration within a designated safe zone and penalising speeds that are either excessively slow or dangerously fast.

2.4 Steering Reward

The "steering reward" function is integral in promoting smooth and responsive steering actions, crucial for the vehicle's safe maneuvering and for following the direction of the road. It calculates rewards based on the angular difference between the vehicle's current direction and the road's trajectory. By adjusting rewards, this function encourages minimal steering angle deviations, ensuring the vehicle follows the road with greater accuracy and stability.

2.5 Curve Deceleration Reward

The "deceleration reward" function is designed to enhance the autonomous vehicle's safety and performance as it approaches curves. By evaluating the vehicle's speed in relation to upcoming road curvatures, this reward system incentivises timely deceleration, helping prevent loss of control or potential accidents on curves. In addition, the vehicle's observation contains information about upcoming curves, which is essential in reducing the speed in advance.

2.6 Stability Reward

The "stability reward" function iterates the importance of maintaining a relatively consistent speed, contributing to a smoother, more predictable driving experience. By comparing current and previous speeds, this function rewards minimal accelerations, whether positive or negative, encouraging the autonomous system to avoid sudden accelerations or deceleration that could cause passenger discomfort or lead to inefficient fuel consumption.

3 Designing and Synthesizing Advanced Policies

3.1 Soft Actor Critic

3.1.1 Algorithm Overview

SAC is predicated on the concept of entropy regularisation. This approach leverages the balance between the maximization of expected reward and the entropy of the policy, which is crucial for effective exploration and exploitation in complex environments. SAC employs an actor-critic architecture, allowing simultaneous learning of a policy and a value function. The policy parameters are optimized via stochastic gradient ascent to achieve a high entropy-enriched reward. This method explicitly trades off exploitation and exploration:

- **Exploitation:** Maximizing the rewards of the "original" problem.
- **Exploration:** Maximizing the policy's entropy (i.e., randomness).

SAC trains "twin delayed" Q-functions, a technique similar to that used in the twin-delayed DDPG (TD3) algorithm. It uses both of these Q-functions for fitting a policy network, which helps improve stability by addressing the issue of Q-function often over-estimating the true Q-value.

3.1.2 Entropy-Regularized MDP Formulation

The entropy-augmented objective function for a policy π in the entropy-regularized Markov Decision Process (MDP) framework of SAC is formulated as follows:

$$\pi^* = \underset{\pi}{\operatorname{argmax}}\, \mathbb{E}_{\tau\sim\pi}\left[\sum_{t=0}^{\infty} \gamma^k \left(r(s_t, a_t) + \alpha\mathcal{H}(\pi(\cdot|s_t))\right)\right] \tag{1}$$

In this expression, π^* denotes the optimal policy that is being sought. The expectation $\mathbb{E}_{\tau\sim\pi}$ is taken over the trajectory τ, which is sampled according to the policy π. The discount factor γ weighs the rewards and entropy over time, $r(s_t, a_t)$ represents the reward at time t, α is the temperature parameter that scales the importance of the entropy term, and $\mathcal{H}(\pi(\cdot|s_t))$ is the entropy of the policy at state s_t. This objective encourages the policy to balance the exploration-exploitation trade-off by maximizing both the policy's cumulative reward and entropy.

3.1.3 Policy and Value Function Updates

The SAC algorithm updates the policy and value functions based on the sampled experiences. The value function is updated to minimize the difference between the predicted Q-values and the target values, which are estimated using the rewards and the entropy-regularized value of the next state. The policy is updated to maximize the expected return along with the entropy, promoting a balance between taking rewarding actions and exploring the action space.

$$\text{Value function update: } V(s_t) \leftarrow \mathbb{E}_{a_t\sim\pi}\left[Q(s_t, a_t) - \alpha \log \pi(a_k|s_k)\right] \tag{2}$$

$$\text{Policy update: } \pi(\cdot|s_t) \leftarrow \underset{\pi}{\operatorname{argmax}}\, \mathbb{E}\left[Q(s_t, \cdot) - \alpha \log \pi(\cdot|s_t)\right] \tag{3}$$

The actor (policy network) and the critic (value function network) are parameterized by neural networks, and the parameters are updated through back propagation and stochastic gradient descent methods.

3.1.4 Soft Q-Function Update

SAC incorporates the soft Q-function, which is adjusted to account for the entropy of the policy. The soft Q-function is updated by minimizing the mean squared error between the predicted Q-values and the target Q-values, which are calculated using the Bellman equation with an entropy term.

$$\text{Soft Q-function update: } Q(s_t, a_t) \leftarrow r(s_t, a_t) + \gamma\mathbb{E}_{s_{t+1}\sim\mathcal{T}}\left[V(s_{t+1})\right] \tag{4}$$

The soft Q-function helps stabilise the training process by smoothing out the value estimates and preventing the overestimation bias common in Q-learning algorithms.

3.1.5 Temperature Parameter Adjustment

The temperature parameter α is critical in balancing the trade-off between exploration and exploitation. SAC includes a mechanism to adjust α automatically based on the entropy of the policy. The objective is to maintain the policy's entropy around a target value, which can be a hyperparameter set by the user.

$$\text{Temperature update: } \alpha \leftarrow_{\alpha} -\mathbb{E}_{a_t \sim \pi} \left[\alpha \log \pi(a_t|s_t) + \alpha \mathcal{H}_{\text{target}} \right] \tag{5}$$

This adjustment allows the algorithm to dynamically balance exploration and exploitation during training, depending on the complexity of the task and the uncertainty in the environment.

3.1.6 Reward scaling and Clipping

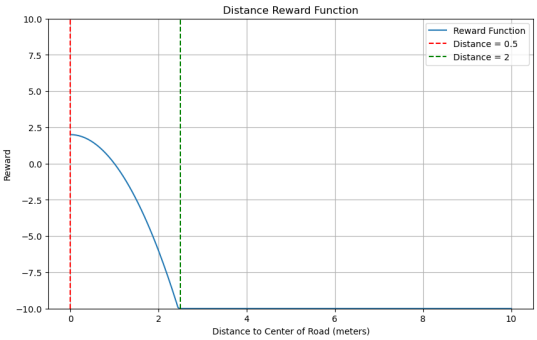
In the Soft Actor-Critic algorithm, reward scaling and clipping play a crucial role in maintaining the stability of the learning process. To achieve this, we apply a smooth reward transition function defined as follows:

```
def smooth_reward_transition(input_reward):
    return np.tanh(input_reward)
```

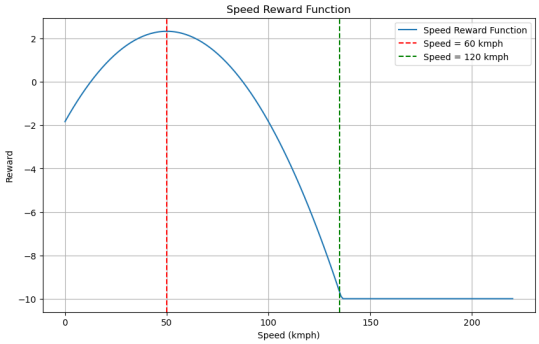
This function employs the hyperbolic tangent (Figure 2f) to scale the input reward, effectively bounding it between -1 and 1. The tanh function serves multiple purposes:

- It normalizes the reward to a consistent scale, aiding in numerical stability.
- The bounded derivative of tanh prevents extreme changes in the policy or value function updates, which could arise from large reward magnitudes.
- It encourages gradual learning and adaptation by the agent, avoiding abrupt policy changes in response to potentially noisy or variable rewards.

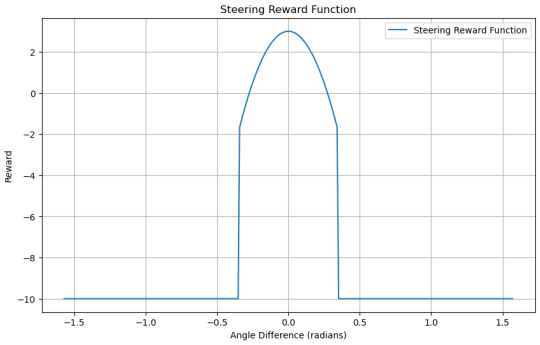
- **Distance Reward (Figure 2a):** The Distance Reward designed to incentivise optimal road centring. It awards the highest reward when the vehicle is precisely at the road's centre, marked by a red line in Figure 2a, symbolizing the ideal position. The reward diminishes parabolic as the vehicle strays from this central point, reflecting the increased navigational risk or inefficiency. This reward reduction is visually represented, reaching zero at a 2.5m deviation from the centre, indicated by a green line.
- **Progress Reward:** Progress reward is determined by the difference in progress between consecutive time steps. This reward is maximised to promote the vehicle to drive forward. The reward is clipped and +10 and -10 to avoid overwhelming other rewards. i.e. The reward is 0 when vehicle is stationary, +10 when it is progressing very fast along the road, and -10 when driving in opposite direction to the destination.



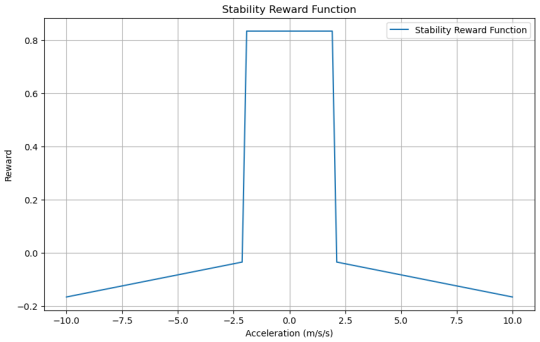
(a) SAC Distance Reward Function



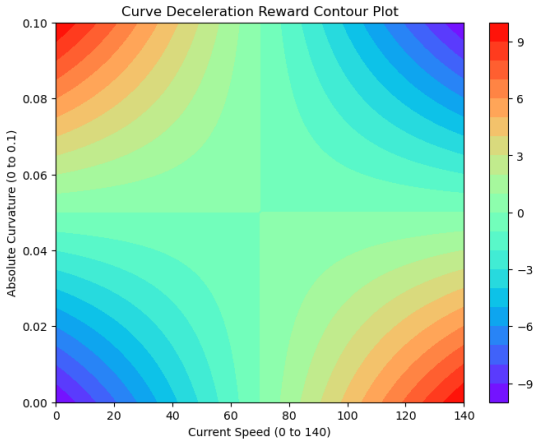
(b) SAC Speed Reward Function



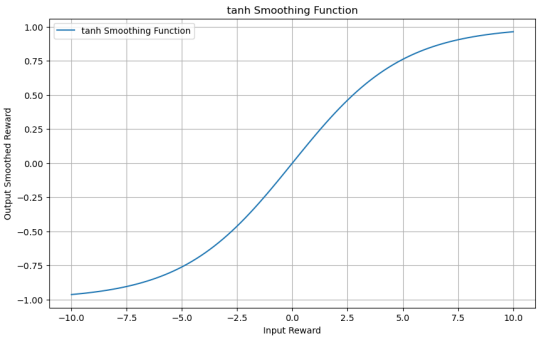
(c) SAC Steering Reward Function



(d) SAC Stability Reward Function



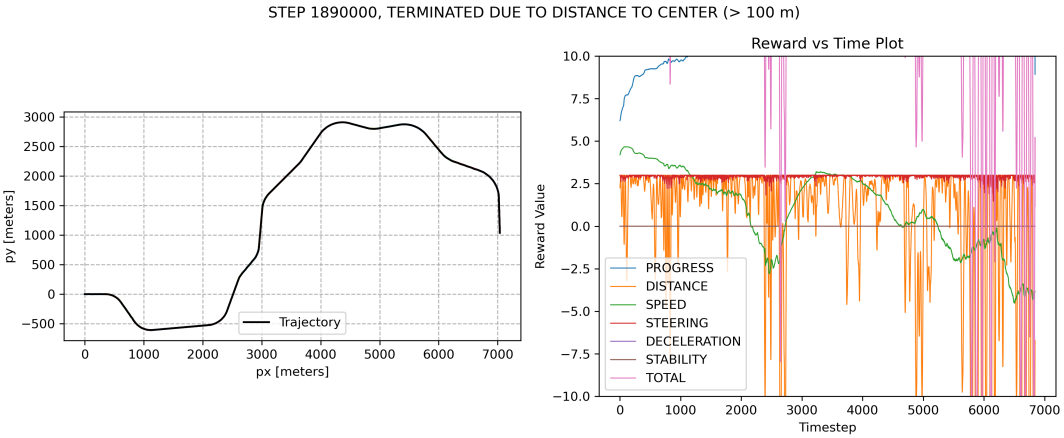
(e) SAC Curve Deceleration Reward Function



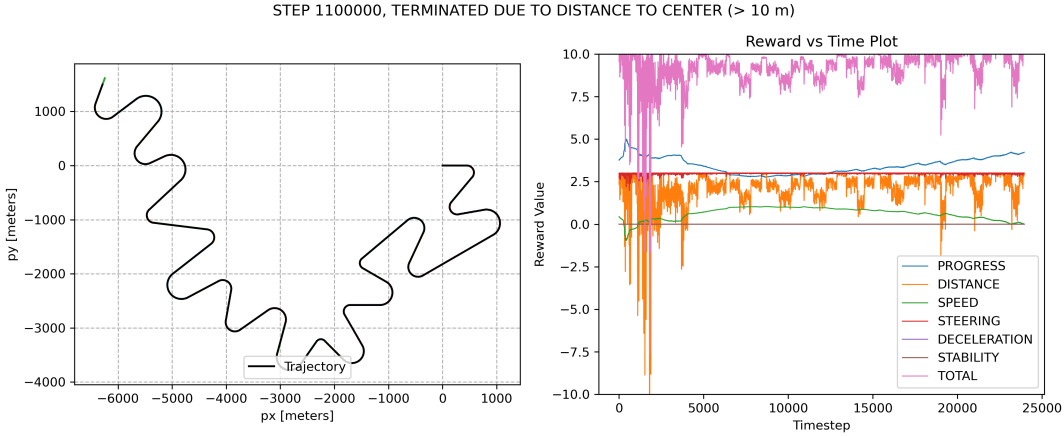
(f) SAC Reward Smoothing Function

Figure 2: SAC Reward Functions

- **Speed Reward: (Figure 2b)**The Speed Reward function ensures it remains within a safe and efficient range. This reward function is mathematically defined as $-1 \times 10 \times (\text{speed} - 50)^2 + 14000$ (The final output of the reward will be rescaled at the end), where the speed is measured in kilometres per hour. The function is designed to peak at 50 km/h (marked by a red line), a threshold identified as the optimal balance between safety and travel efficiency. As the vehicle's speed deviates from this optimal point, either by slowing down or speeding up, the reward decreases quadratically, indicating a less desirable state. This steep quadratic



(a) SAC Training with Small Map, Trajectory and Reward Plot

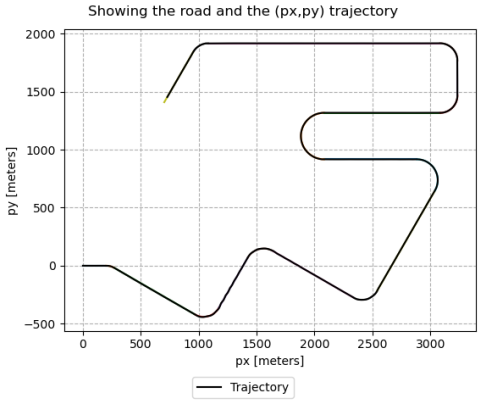


(b) SAC Training with Large Map, Trajectory and Reward Plot

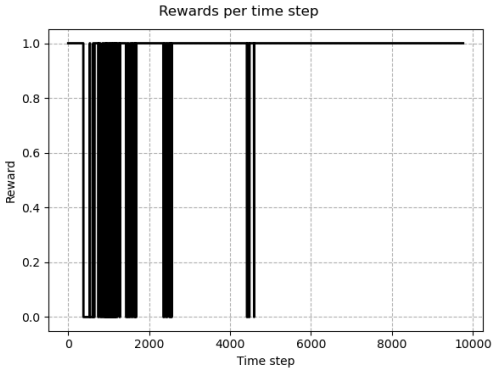
decrease emphasises the importance of adhering closely to the 50 km/h speed. Notably, the reward drops to zero at approximately 140 km/h, a threshold marked with a green line. This sharp decline to zero rewards at higher speeds is a strong deterrent against excessive speeding, aligning the vehicle's behaviour with safety standards.

- **Steering Reward (Figure 2c):** Steering reward ensures the vehicle attempts to orientate itself in the direction of the road, and penalises it otherwise. From Figure 2c we can see that the steering reward is maximum at 0 degree delta between the vehicle and the road. This is the ideal condition. The reward decreases slightly when the vehicle deviates from this, promoting it to be centred but allow small deviations which occurs naturally at corners. However, at around 25 degree, which we have designed as our cut off angle that anything greater than this will attract a large penalty. Having the vehicle deviate this much from the road is not desirable and not safe in real world, especially at high speeds.

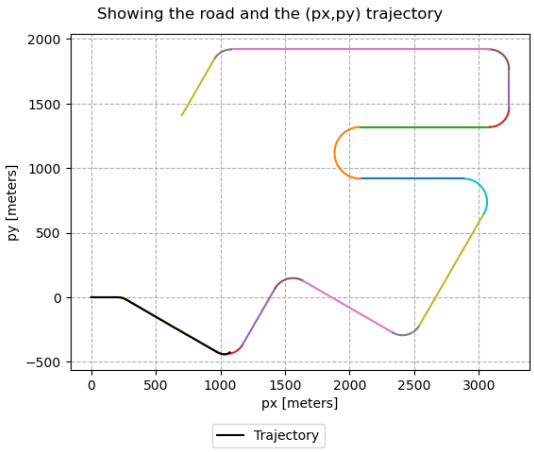
- **Stability Reward (Figure 2d:** The stability reward rewards the vehicle for having minimum acceleration and deceleration. This is mainly for passenger comfort and fuel efficiency, rather than for vehicle safety. For example, having a high deceleration will make the car brake faster which is not necessarily bad. However, it is



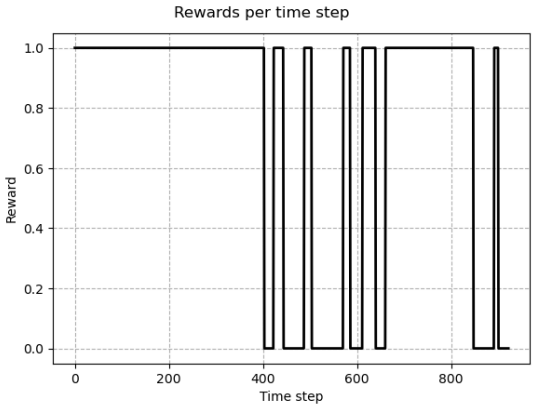
(a) SAC Evaluation with Testing Map, Trajectory, Success



(b) SAC Evaluation with Testing Map, Reward, Success



(c) SAC Evaluation with Testing Map, Trajectory, Fail



(d) SAC Evaluation with Testing Map, Reward, Fail

Figure 4: SAC evaluation on different map

included and will be scaled so that it contributes to the reward but not as much as the other safety critical aspects. For our reward function, we have determined to keep the car's acceleration below 2 m/s^2, which attracts a positive reward. Values outside this will attract a tiny negative reward to not penalise large accelerations too much.

- **Curve Deceleration Reward (Figure 2e:** This reward is designed to promote preemptively slowing down at curves and speeding up at straight roads. The reward function is based on two parameters as shown in Figure 2e. The plot is shown as a contour map, with red being a higher reward and purple being a lower one. The vehicle looks ahead and determines if there are any curves 100 metres ahead. And uses the curvature ahead and its current speed to determine the reward. A high speed and high curvature ahead will definitely attract large penalty. Similarly, a low speed on a straight line will also attract penalty. Where as slowing down when approaching high curvature corners will gain reward. Speeding up at straight roads will also gain reward.

- **Scaling and Weighting:** The scaling and weighting of rewards are crucial for

effective learning and performance optimisation. The process involves using empirically determined "magic numbers", such as the 14000 in the speed reward, which is fine-tuned through numerous iterations of testing and training. This iterative process is guided by visualising the outcomes during training sessions, allowing for adjustments in the weights of different rewards based on their impact on the vehicle's behaviour. For instance, the weights of the deceleration and stability rewards were reduced after observing their relative importance in the training process. In contrast, the distance and speed rewards are given higher weights as they motivate the vehicle to follow the road accurately and maintain an optimal speed. Similarly, the steering reward is crucial for teaching the vehicle to navigate turns effectively. Another critical component is the progress reward, which has been observed throughout hundreds of tests, to be essential in initiating movement. The vehicle remains stationary at the starting point without a sufficiently high progress reward. Therefore, these rewards are assigned higher weights than others, reflecting their significance in achieving the desired driving behaviour and learning outcomes.

3.1.7 Observations

- **Distance to Road:** This observation measures the closest distance of the car from the centre of the road. It's crucial for the vehicle to understand its lateral position relative to the road's centre line, aiding in maintaining lane discipline.
- **Angle Difference:** This captures the angular deviation between the car's heading and the road's tangent at the closest point. It's essential for the vehicle to align its direction with the road's curvature, especially in turns.
- **Curvature at Closest Point:** Reflects the road's curvature at the point nearest to the car. This observation helps the vehicle anticipate and adapt to upcoming road bends.
- **Progress at Closest Point:** Indicates the car's progress along the track at the closest point. This metric is vital for understanding how far the car has travelled and its position on the track.
- **Velocity:** The car's current speed combines the x and y components. This observation is crucial for speed control and ensuring the vehicle moves at an appropriate pace.
- **Angular Velocity:** Measures the rate of change of the car's orientation, which is key for understanding how quickly the car turns and adjusting steering accordingly.
- **Look-Ahead Curvature:** This observation predicts the road's curvature at a certain distance ahead of the car. It allows the vehicle to prepare in advance for future road conditions, enhancing its ability to navigate turns and bends smoothly.
- **Not included in observation:** Information such as the absolute position of the car, or the absolute orientation of the car is not included. These are not as useful as the other observations above. Having too much irrelevant information available to the policy is likely going to confuse the policy, when finding correlation between a specific action, reward, and observation.

3.1.8 Training

The training of the algorithm was done on two different sets of maps. Firstly it was done on a smaller and easier map with less sharp corners and shorter length in total as shown in Figure 3a. This is to do a preliminary testing on whether our policy synthesis has worked, before devoting more computational resources and time to train a more comprehensive model, shown in Figure 3b.

Training was done on a local computer running Ubuntu 22.04. Multiple algorithms such as PPO and SAC were run simultaneously to utilise the resources available. The training machine was upgraded with Intel i9 13900KS with 64 GB of RAM, as well as a Nvidia RTX 4090 with 24 GB of VRAM. Both CPU and GPU were utilised for training. The machine is running 24/7 with a script to automatically upload the results to a gitlab server for access at any time even at uni. Power consumption was a concern and it was heating up the house pretty easily. This is a trade off between training online and offline. A policy such as MPC will take more resource on the machine running the algorithm, where as RL such as SAC takes considerable amount of computational power for training, but requiring little for realising it.

One of the ways we adjusted for the rewards is to test many runs on smaller maps, and updating the policy based on the result. For the smaller map in Figure 3a, the policy completed the training with relative ease. On the left is the trajectory and the right is individualised reward vs time plot. From the reward plot we can see the distance to centre penalty may be too aggressive that it is oscillating. Other than that the steering angle seems reasonable. But the progress is giving a very high reward, with speed reward declining. This indicates that the vehicle is possibly speeding.

The reward functions and weightings are updated accordingly for the larger map in Figure 3b. The map was able to be completed successfully with a stable progress, speed, steering reward. The distance reward is not as varied as before but seems reasonable as there are lots of curves and possibility to deviate from the road at those curves.

The training and updating of the parameters were repeated for hundreds of times to obtain a working version of the policy using SAC.

3.1.9 Evaluation on testing road

The evaluation of the SAC model's performance was conducted on a large scale of testing maps with the following criteria for reward assignment:

- A zero reward is assigned if the vehicle maintains a minimum distance of 1 meter from the edge of the road.
- Conversely, a reward of one is given for any deviation beyond this 1-meter threshold.

After rigorous testing, the well-trained SAC model demonstrated a 70% success rate in completing the designated road courses, as illustrated in fig 4b. However, it was observed that the model occasionally failed fig 4d. This can be attributed to the stochastic nature of the SAC. The SAC incorporates stochastic policies that can result in different actions even in similar states, which may occasionally lead to failures.

3.2 Proximal Policy Optimisation

3.2.1 Algorithm Overview

PPO is an advancement over previous policy optimization methods due to its balance between efficiency and ease of implementation. It operates on the principle of making the most significant possible improvement without causing harmful large policy updates. PPO's objective function is carefully designed to prevent destructive large updates through a clipping mechanism, ensuring smooth policy evolution. The core components of PPO are:

- **Policy Update Constraint:** Ensures that the policy does not deviate excessively from the previous policy.
- **Clip Mechanism:** Modifies the objective function to clip the policy update at each iteration.

PPO retains the advantages of policy gradient methods while simplifying the implementation and reducing the computational burden compared to its predecessors. The simulations and training complete in a considerable shorter time than other algorithms such as SAC.

3.2.2 Clipped Objective Function

The clipped surrogate objective function in PPO, which aims to keep policy updates within a safe range, is given by:

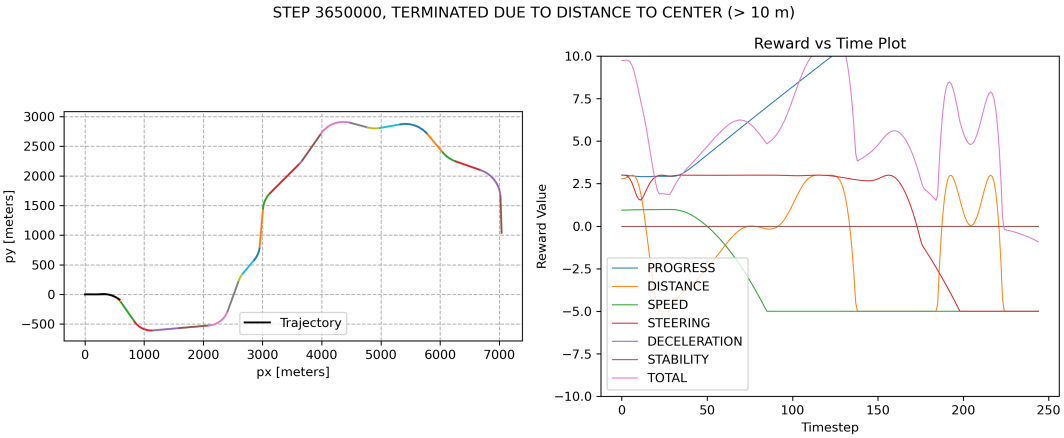
$$L^{CLIP}(\theta) = \hat{\mathbb{E}}_t \left[\min(r_t(\theta)\hat{A}_t, \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t) \right] \tag{6}$$

Where $L^{CLIP}(\theta)$ denotes the clipped surrogate objective function, $\hat{\mathbb{E}}_t$ represents the empirical expectation over a finite batch of samples, $r_t(\theta)$ is the ratio of the probability under the new policy to the probability under the old policy for taking action a_t in state s_t, $\hat{A}_t$ is an estimator of the advantage at time t, and ϵ is a hyperparameter that defines the clipping range. This objective function is central to PPO's approach of making conservative policy updates.

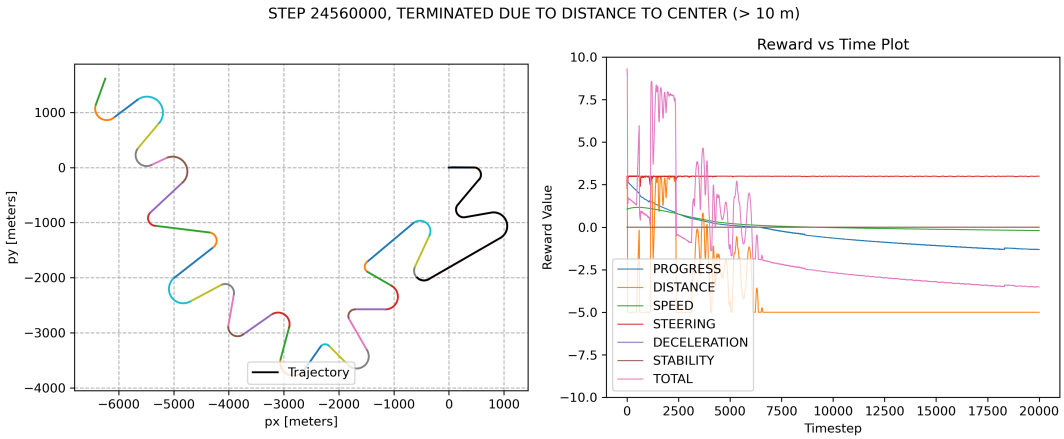
3.2.3 Reward Scaling and Clipping

In our PPO implementation, we continued using the reward structure established in our SAC model, which encompasses essential aspects of autonomous driving, such as speed, steering, progress, etc. We believe these rewards effectively mirror the feedback required during driving tasks. A key modification in our approach was the removal of the tanh function 3.1.6 and introducing of the clipping function for reward scaling. The clipping function is defined as:

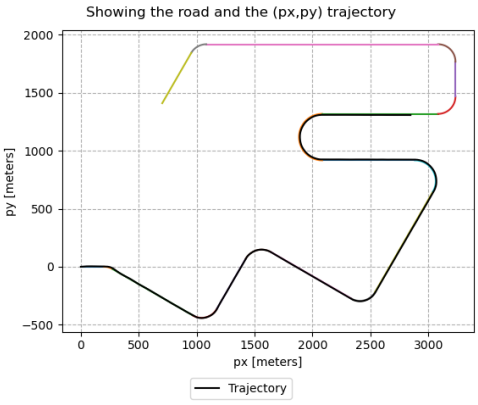
$$\text{clip\_reward\_range}(\text{input\_reward}, \text{max} = 1000) = \begin{cases} \text{max} & \text{if input\_reward} > \text{max} \\ -\text{max} & \text{if input\_reward} < -\text{max} \\ \text{input\_reward} & \text{otherwise} \end{cases} \tag{7}$$



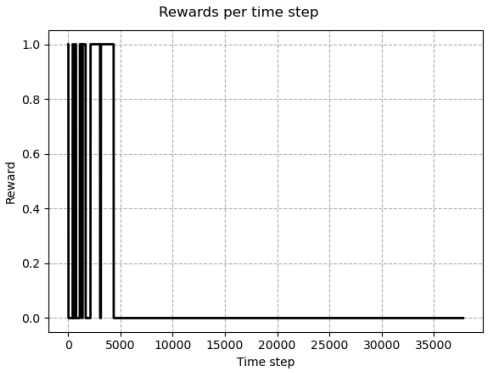
(a) PPO Training with Small Map, Trajectory and Reward Plot



(b) PPO Training with Large Map, Trajectory and Reward Plot



(a) PPO Evaluation with Testing Map, Trajectory



(b) PPO Evaluation with Testing Map, Reward

Figure 6: PPO evaluation on different map

This adjustment was made to mitigate the issue of extreme reward values, which can lead to instability in policy updates. By capping the rewards at a maximum and minimum threshold, we aim to balance the learning process, ensuring that the agent is neither overly penalised nor excessively rewarded. This method helps to stabilise the

training process, making it more robust against the potential pitfalls of erratic reward distributions.

3.2.4 Observations

In our Proximal Policy Optimization implementation, we have decided to utilise the same set of observations as previously employed in Subsection 3.1.7. This decision is based on the comprehensive nature of the existing observations, which adequately capture all necessary aspects for effective autonomous driving. Collectively, these observations provide a robust framework for the PPO algorithm to learn effective driving policies, encompassing key aspects such as positioning, alignment, and vehicle dynamics.

3.2.5 Training

Similar to SAC, the training of the PPO algorithm was done on two different sets of maps. Firstly it was done on a smaller and easier map with less sharp corners and shorter length in total as shown in Figure 5a. Then the training is done on a larger map again for more complex cases. Figure 5b.

Similarly, training was done on the same Linux computer as SAC, since it had enough resources to run both algorithms in parallel. Multiple python notebooks were open and specific values such as reward and observations were adjusted individually in each notebook.

During the training for the PPO algorithm, the training did not finish all the way to the end, as shown in Figure 5b. Unfortunately we have set the maximum time step to be 20000, which was not enough to complete the entire map. We ran the training repeatedly but only realised it 6 hours before the report is due. We did not have enough time to run another training with larger time step, as it takes at least 12 hours to generate anything useful. It took 24,560,000 time steps to come up with the current model that seems usable. Therefore, further evaluations are performed using this model in Figure 5b.

From the reward plot on the right of Figure 5b, we can see that the progress reward for this set of training model is very low. Indicating the speed of the vehicle is low. This is expected with PPO trying to be as stable as possible. The steering angle delta reward is pretty much always at maximum, indicating a vehicle that closely follows the direction of the road. However, the model has a consistently low distance to centre reward, indicating it is trying to follow the line, but instead of driving in the centre, it drives slightly offset from the centre. This may be caused by the reward not penalising slight deviations.

Similar to SAC, the parameters were updated hundreds of times across runs to come up with the final parameters.

3.2.6 Evaluation on testing road

In Figure 6a and Figure 6b, the results for the evaluation and reward metrics are shown using the model in the section above. The model is able to complete 70% of the road before deviating too much to be stopped. However, by looking at the training model, the distance to the centre is pretty offset from the centre. Which resulted in a low reward metric in Figure 6b. Where the vehicle rarely stays within 1 metre of the centre of the line. Indicating that PPO is stable that in different evaluation environments, the result is extremely predictable. Evaluations with the same map was repeated multiple times. Each time the outcome is consistent with only very small differences between evaluation

runs. This further proves that PPO is stable that it is more suitable in safety critical systems such as autonomous driving. We have performed the same simulation 20 times, and not a single simulation failed catastrophically.

4 Comparing Policies

4.1 Key Performance Metrics

- **Safety:** In our tests, SAC demonstrated a 70% success rate in navigating the testing road, indicating a moderate level of reliability. However, it showed variability in its performance, which could be a concern in real-world applications where consistency is crucial for safety. On the other hand, PPO, despite not completing the road in our trials, exhibited a high degree of stability in its performance. The outcomes were consistent across iterations, often ending at the same location. This predictability, despite the lower success rate, might be more desirable in real-world scenarios. The consistent behaviour of PPO, even if it means not completing the route, could be preferable for a real car's reinforcement learning algorithm, as it reduces the unpredictability and potential safety risks associated with more variable behaviours. Hence, we can conclude that PPO outperforms SAC due to its stability.

- **Efficiency:** In terms of efficiency, characterised primarily by the overall speed of the vehicle, SAC outperforms PPO. SAC exhibits a higher average speed, suggesting a more efficient traversal of the test route. This could indicate SAC's ability to balance speed with other driving factors effectively. In contrast, PPO maintains a stable but slower speed throughout its operation. While this may contribute to safety and stability, as noted earlier, it does so at the cost of efficiency. PPO's conservative approach to speed may lead to longer travel times, which could be less efficient when time is a critical factor.

4.2 Trade-offs and Multi-Objective Analysis

- **Pareto Efficiency:** For autonomous driving, Pareto efficiency is critical in the context of balancing efficiency and safety. For example, with our policy choices, SAC and PPO, we achieve better safety with a more stable simulation with PPO. However, PPO is less efficient during training and took longer to complete. The vehicle also travelled slow with PPO. Where as SAC had higher probabilities in crashing in our simulations, but completed much more efficiently due to its high entropy nature.

- **Pareto Dominance:** Pareto dominance is when a policy is as good as another policy, if not better. Which means there is no need to use the policy that is dominated by the other policy. When we choose the policy to use, we chose SAC and PPO. Not only they differ in their algorithms, they are also "the best in their variants". For example, SAC is an "upgraded" version of DDPG, with most aspects being the same if not better than DDPG. Therefore, it eases our choices of policies by choosing the "best" policy.

- **Pareto Front:** The Pareto Front illustrates the inherent trade-offs between SAC and PPO. SAC excels in controlled, predictable environments, while PPO is advantageous in unpredictable or hazardous conditions. This analysis aids in selecting the appropriate policy based on specific operational requirements and priorities, highlighting the conditions under which each policy performs best.

4.3 Expected vs. Observed Performance

- The comparison between the expected and observed performance of SAC and PPO algorithms reveals insightful discrepancies. Initially, SAC was anticipated to excel in efficiency and safety due to its entropy regularisation approach, which theoretically balances exploration and exploitation. However, in practice, while SAC showed superior efficiency, its safety performance was less consistent than expected, with a success rate of only 70% in navigating the testing road. This variance highlights the challenge of translating theoretical advantages into real-world scenarios.

  On the other hand, PPO, known for its stability and simplicity, was expected to demonstrate moderate performance across all metrics. Surprisingly, its observed safety performance was notably higher than anticipated despite a lower overall success rate. PPO's consistent behaviour, even in failing to complete the route, suggests a high level of predictability and reliability, which is crucial for safety in autonomous driving. This contrast between expected and observed performances underscores the importance of empirical testing in complex environments, where theoretical models may not fully capture the nuances of real-world dynamics.

5 Reflection and Learning Outcomes

5.1 Reflection 1 on Sam Kirkpatrick from ANCA

Sam Kirkpatrick's exposition on Model Predictive Control in the industrial settings shows the complex relationships between theories and their applications. Reflecting on his statement, "We are sort of establishing, I guess, world-first trends in how these sort of things look in the real world", we can appreciate the effort spent to bridge the gap between MPC as a theoretical topic and its practical deployment in an industrial setting.

This quote reiterates the idea that implementing MPC goes beyond academic environments and how it is used in industries. It shows a process of innovation, where the theoretical models of MPC are put to the test against the unpredictable variables of real-world environments. As someone who has explored into the world of MPC in this class, this transition from theory to practice exposes critical challenges. It demonstrates the need to validate the robustness of MPC models in the face of real-world constraints, such as equipment tolerances and the variability inherent in environment and production processes.

Kirkpatrick's reflection on MPC's application is an important reminder that theoretical models must be adaptable and resilient to the complexities of their physical operational contexts. In practice, this means not only carefully designing the predictive models to take into account a range of potential scenarios, but also ensuring that the system remains flexible and responsive to sudden changes. The quote suggests that the real measure for MPC in industry is not only by how well it performs within simula-

tions, but also by how it can be integrated into the operational workflows and maintain performance and safety over time.

Relating MPC's place within industry, Kirkpatrick's insights highlight the importance of learning the theory of MPC, but more in using MPC as a tool to tackle the complexities and demands of modern industrial systems.

5.2 Reflection 2 on Sam Kirkpatrick from ANCA

Reflecting on the practical applications of MPC, Sam Kirkpatrick's words, "It's not really in the name, predictive was in the name. It's control and it's gonna have a model. But what's not maybe clear from the name is that it's very much an optimisation based method", which identifies the core of MPC's functionality and the common misconceptions about it. This quote shows that it's fundamentally about optimisation, something that might not be immediately obvious to those unfamiliar with MPC.

Kirkpatrick's words serve as an important reminder of the optimisation nature of MPC, which is crucial in an industrial context. MPC is not just about prediction, it is about using those predictions in an optimisation to control a system in an optimal way. This is what makes it important in industry, where decisions must take into account the future states and the feasibility of these states.

It is evident that the challenges of implementing control systems in industry are not only technical but also conceptual. We must identify that while MPC can predict future states based on a model, its real uses lies in its ability to use predictions to optimise control actions.

The lecture reinforces the idea that the predictive capabilities of MPC are methods to achieve desired outcomes. This is critical when considering the use of MPC in manufacturing and other industrial processes. It's not just about understanding the system, it is also about understanding how to use it efficiently.

In our MPC implementation, it's clear that understanding the optimisation based nature of MPC is critical. It is this understanding that connects the theoretical models to their practical application, enabling engineers to design MPC systems that are not only predictive, providing control strategies that align with physical objectives and realities. This view of MPC as an optimisation tool is essential for its successful application in the complex environment of industrial automation.

5.3 Reflection 3 on Michael Crump from BAE Systems

The application of AI for Robotics in defence signifies a paradigm shift where machines are tasked with not just executing predetermined actions but also making autonomous decisions in critical situations. Michael Crump from BAE Systems elucidated this transition, highlighting how robotics with autonomous navigation and target identification has progressed from theoretical frameworks to real-world applications. His statement, "The frontier of defence technology is increasingly being shaped by the intelligent systems we deploy," reflects this shift towards autonomous capabilities. This evolution introduces unique challenges, such as the unpredictability of combat environments, the necessity for instantaneous decision-making, and the ethical considerations of autonomous operations. These challenges are precisely what we encounter in the field of AI and what engineers must strive to solve. As AI systems gain autonomy, the complexity of their operational environment demands robust algorithms capable of adapting to unforeseen events. The

need for swift decision-making in AI also calls for real-time processing capabilities and fail-safe protocols. Furthermore, the ethical implications of autonomous machines in defence require a thoughtful approach to engineering, ensuring that the technology we develop aligns with our societal and moral standards.

In our AI4R course, we've engaged with these concepts at a foundational level, conducting extensive simulation projects on autonomous driving and testing algorithms on scaled-down car models. These exercises mimic real-world scenarios, allowing us to appreciate the complexities Crump described—especially when our simulations encounter unexpected variables that test the robustness of our programming.

The integration of AI in defence, as Crump notes, brings forth intricate ethical questions about the autonomy of machines and the rules of engagement. This underscores the crucial role of engineers in developing AI solutions responsibly, with a broad understanding of their societal and combat implications. Our classwork, reflecting this, not only focuses on the mechanics of AI but also its potential consequences, preparing us to design systems that are both effective and ethically sound.

5.4 Reflection 4 on Peter Cudmore from Arkeus

Reflecting on Peter Cudmore's talk on machine vision, the importance of machine learning in real-world environments was profoundly emphasised, particularly highlighting the challenges in data acquisition and handling data scarcity. This ties closely with our coursework on reinforcement learning and Markov Decision Processes, where we focus on optimising algorithms in situations with limited data availability.

A vital aspect of our learning was illuminated through our experiments in the Sim-2-real project. The project demonstrated the critical role of machine vision, especially under varying conditions. We observed firsthand how noise and different camera settings significantly impact the performance of our models. For instance, varying lighting conditions during our simulations led to noticeable performance variations in our autonomous car models. This real-world experimentation underlined Peter Cudmore's point about balancing perception and awareness in autonomous systems. He asserted, "If you have a data set that doesn't capture the features that you care about at higher enough resolution, then you won't be able to interpolate between those features when you deploy that in the real world." highlights the challenge of creating machine learning algorithms that are not just efficient with data but can also generalise from limited data sets, a principle that aligns with our goal of building robust models capable of adapting to new and unforeseen scenarios.

Peter Cudmore's discussion on the limitations and potential pitfalls of machine learning systems in real-world applications further resonates with our experiences. He stressed integrating machine learning with traditional computer vision techniques and control theory for more reliable and robust autonomous systems. This multidisciplinary approach mirrors what we've practised in our course projects, like the autonomous driving simulations, where the unpredictability of real-world conditions often tests the resilience of our programming.

References

[1] Schulman, J., Wolski, F., Dhariwal, P., Radford, A., & Klimov, O. (2017). Proximal Policy Optimization Algorithms. arXiv preprint arXiv:1707.06347.

[2] OpenAI. (n.d.). Soft Actor-Critic — Spinning Up documentation. Retrieved November 13, 2023, from https://spinningup.openai.com/en/latest/algorithms/sac.html