# Vision-Based Regularizing Action Policies for Smoothing Control in Autonomous Miniature Car Racing

Bo-Jiun Hsu[1], Hoang-Giang Cao[1,2], I Lee[1], Chih-Yu Kao[1], Jin-Bo Huang[1], and I-Chen Wu[†1,2]

[1]*Department of Computer Science, National Yang Ming Chiao Tung University, Taiwan*
[2]*Research Center for IT Innovation, Academia Sinica, Taiwan*

## Abstract

Deep reinforcement learning has achieved significant results in low-level controlling tasks. However, for some applications like autonomous driving and drone flying, it is difficult to control behavior stably since the agent may suddenly change its actions that often lower the controlling system's efficiency, induce excessive mechanical wear, and cause uncontrollable, dangerous behavior to the vehicle. Recently, a method called Conditioning for Action Policy Smoothness (CAPS) is used to solve the problem of jerkiness in low-dimensional features for applications such as quadrotor drones. In this paper, we extend CAPS to image-based CAPS and also apply it to solve jerky control in autonomous miniature car racing. Our experiments show that combining CAPS and sim-to-real transfer methods helps stabilize the control at a higher speed. Especially, the agent with CAPS and CycleGAN reduces the average finishing lap time, while also improving the completion rate. We also conduct extensive experiments to analyze the impact of CAPS components.

## 1 Introduction

In recent years, deep reinforcement learning (DRL) has been applied to many real-world applications and achieved many milestones. However, a critical problem in control policy is the jerky behavior, when training the agent in a complex dynamical environment [7; 8]. Especially, in autonomous driving or quadrotor drone flying, jerky control causes many serious problems, such as uncontrollable moving and power-consuming, which reduce the service life of the autonomous vehicle. Prior works addressed the issue of smoothness policy by using reward engineering [5; 6]. This approach designed a reward function for a specific task. In autonomous driving, for example, the agent will be penalized if the current action is too different to the previous action, or the selected speed is too slow. Reward engineering is based on prior human knowledge about the tasks. It is easy to implement, but hard to design a good reward function. Recent researches used DRL algorithm to solve this problem, trying to maximize total episode reward, and also smoothing control or action oscillation. Yu at el. presented temporally abstract actor-critic TAAC, an off-policy reinforcement learning incorporates with closed-loop action repetition (temporal abstraction)[10]. A work in [2] proposed Nested Soft Actor-Critic (NSAC), a DRL algorithm that helps to reduce oscillation behavior in autonomous driving compared to a range of commonly adopted baselines with almost similar performance. Siddharth Mysore et al. proposed Conditioning for Action Policy Smoothness (CAPS) for solving jerky actions by adding regularization terms[9]. CAPS was originally applied to smooth the control of quadrotor drones with some low dimensional features.

**Contributions.** The main contributions of this paper can be summarized as follow: 1) We extend CAPS[9] to image-based CAPS, and apply to solve jerky control in autonomous car racing. 2) In the experiments, we show that CAPS helps stabilize the car when moving at a higher speed. Especially, the method that combines CycleGAN and CAPS outperforms other methods and reduces the average finish lap time, while also improving the completion rate. 3) We also conducted extensive experiments to study the impact of CAPS components.

## 2 Conditioning for Action Policy Smoothness

Siddharth Mysore et al. introduced Conditioning for Action Policy Smoothness (CAPS)[9] and got significant improvements in controller smoothness and power consumption on a quadrotors drone. To condition policies for smooth control, the authors proposed two regularization terms: 1) Temporal Smoothness term. 2) Spatial Smoothness term.

The policy $\pi$ is a mapping function of states $s$ to actions $a = \pi(s)$. The objective function of CAPS, $J_\pi^{\text{CAPS}}$, contains three components: objective function of Soft Actor-Critic $J_\pi$; Temporal Smoothness regularization term $L_T$; and Spatial Smoothness regularization term $L_S$. The regularization weights $\lambda_T$ and $\lambda_S$ are used to balance the impact of two regularization terms $L_T$ and $L_S$, respectively.

$$J_\pi^{\text{CAPS}} = J_\pi - \lambda_T L_T - \lambda_S L_S \tag{1}$$

$$L_T = D_T(\pi(s_t), \pi(s_{t+1})) \tag{2}$$

$$L_S = D_S(\pi(s_t), \pi(s'_t)) \quad where \quad s'_t \sim \Phi(s_t). \tag{3}$$

Both $D_T$ and $D_S$ are calculated based on the Euclidean distance. The Temporal Smoothness term $L_T$ penalizes the $J_\pi^{\text{CAPS}}$

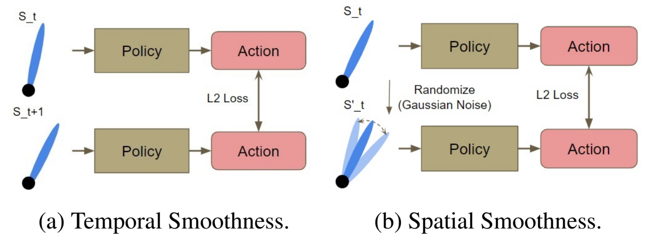


Figure 1: Condition policies for smooth behavior in CAPS.

when the action of the next states $s_{t+1}$ are significantly different from the actions of the current states $s_t$. The Spatial Smoothness term $L_S$ encourages the policy to take the similar actions on the similar states $s_t'$, which are drawn from a distribution $\Phi$ around states $s_t$. Originally, CAPS is applied to smooth the control of a quadrotor drone, with the input being the inertial measurement unit (IMU) and the electronic speed controller (ESC) sensors. Therefore, to generate similar states $s_t'$, the authors sampled from a normal distribution, $\Phi(s) = N(s, \sigma)$ with standard deviation $\sigma$, around $s_t$. Figure 1 illustrates the Temporal Smoothness and Spatial Smoothness in CAPS.

## 3 Our Approach

### 3.1 Image-based CAPS for Autonomous Car Racing

CAPS was originally applied to smooth the control of a drone with the internal states of the rotors. Then, to generate the similar state $s_t'$ in Spatial Smoothness, the authors used Gaussian Noise to draw $s_t'$ from a normal distribution around state $s_t$ as described in section 2. In this paper, we extend the idea of using CAPS with image-based input and then use it to smooth the control of an autonomous miniature car racing. Since our approach use image as the input, therefore, to generate the similar state $s_t'$ in the Spatial Smoothness in CAPS, instead of drawing from a normal distribution, we implement 6 different domain randomization methods [1]: a) Random Brightness. b) Random Contrast. c) Random Rotation. d) Salt and Pepper. e) Gaussian Blur. f) Random Cut-off. See Appendix A for the details of implementation of domain randomization methods.

In the experiment, we will study the impact of CAPS components in Equation 1. We also analyze the sensitivity of regularization weights of the Temporal Smoothness and the impact of domain randomization methods in Spatial Smoothness through the ablation study.

### 3.2 Sim-to-real Transfer for Autonomous Car Racing

Sim-to-real transfer plays an important role in the sim-to-real task. In sim-to-real tasks, we trained the policy in the simulation and then directly apply it to the real-world environment.

To analyze the effectiveness of different sim-to-real transfer approaches in autonomous miniature car racing, we compare CycleGAN [11], a domain adaptation method, with different domain randomization methods.

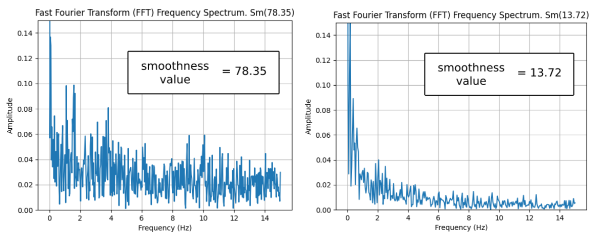


Figure 2: Smoothness value of speed without CAPS (left) and with CAPS (right).

## 4 Experiments

In the experiment, we first verify the performance of image-based CAPS in 4.2. We then combine CAPS with different sim-to-real methods to compare the effectiveness of sim-to-real transfer for autonomous miniature car racing (4.3). Finally, we conduct extensive experiments to analyze the impact of CAPS components (4.4, 4.5, 4.6).See Appendix B for more details about the experiments setup.

### 4.1 Evaluation Metrics

To evaluate the effectiveness of the smoothness method, we use two following metrics:

1. Mean of selected steering angle: measure the angle difference when the agent decides to change the moving direction.

2. Smoothness value: proposed by [9], a method based on the Fast Fourier Transform frequency spectrum as defined in Equation 4 . The lower the value, the higher the low-frequency control and smoother the action.

$$S_m = \frac{2}{n f_s} \sum_{i=1}^{n} M_i f_i \tag{4}$$

where $M_i$ is the amplitude of the frequency component $f_i$, and the $f_s$ is the sampling rate. We set $f_s = 30$ in this work.

To evaluate the performance of the car racing, we use two metrics: (a) Average finishing lap time: the average time to finish a run (in second). (b) Completion rate: the % number of completion runs per all runs.

### 4.2 Smoothness with Image-based CAPS

In this experiment, we study the policy smoothing ability of image-based CAPS. We train the agents in the simulation and test them in the real environment. For the sim-to-real transfer method, we applied CycleGAN for all agents.

Figure 2 to 4 show the comparison of the steering angle and the speed with CAPS and without CAPS. The model with image-based CAPS achieves more stable behaviors in the real-world testing, therefore, significantly improves the smoothness value in both speed and steering angle. More specifically, the smoothness value of steering angle reduces from 93.47 to 20.39; and from 78.35 to 13.72 for smoothness value of speed.

| Setting | Speed range (m/s) | | Baseline | | Domain Randomization | | CycleGAN | |
|---|---|---|---|---|---|---|---|---|
| | Min | Max | w/o CAPS | with CAPS | w/o CAPS | with CAPS | w/o CAPS | with CAPS |
| *Slow* | 1.125 | 8.0 | 0% | 0% | 0% | **100%** | 73.33% | 73.33% |
| *Medium* | 1.125 | 9.0 | 0% | 0% | 0% | 26.67% | **86.67%** | 80.00% |
| *Fast* | 3.000 | 9.3 | 0% | 0% | 0% | 0% | 13.33% | **20.00%** |

Table 1: Completion rate in real-world testing.

| Setting | Speed range (m/s) | | Baseline | | Domain Randomization | | CycleGAN | |
|---|---|---|---|---|---|---|---|---|
| | Min | Max | w/o CAPS | with CAPS | w/o CAPS | with CAPS | w/o CAPS | with CAPS |
| *Slow* | 1.125 | 8.0 | NaN | NaN | NaN | 32.47s | 22.20s | **17.36s** |
| *Medium* | 1.125 | 9.0 | NaN | NaN | NaN | 24.33s | 19.77s | **16.05s** |
| *Fast* | 3.000 | 9.3 | NaN | NaN | NaN | NaN | 17.20s | **14.71s** |

Table 2: Average finishing lap time in real-world testing.

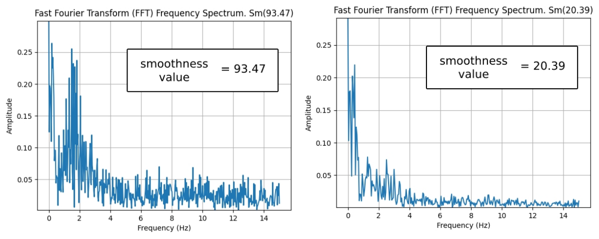
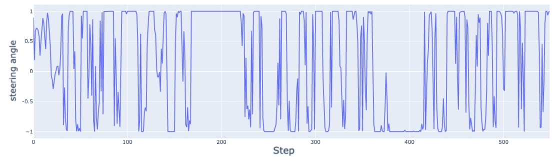
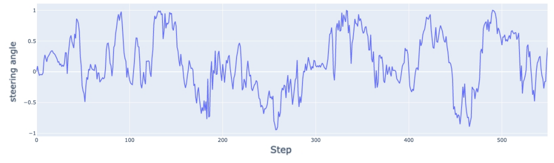


Figure 3: Smoothness value of steering angle without CAPS (left) and with CAPS (right).



(a) Steering angle of the car without CAPS.



(b) Steering angle of the car with CAPS.

Figure 4: Steering angle of the car with and without CAPS.

## 4.3 Smoothness Control and Sim-to-real Policy Transfer

In this experiment, we compare the effectiveness of both the action smoothing method and sim-to-real transfer methods. We implement a version of Ape-X[4] with Soft-Actor-Critic (SAC)[3] without sim-to-real transfer method as the baseline. We then combine different sim-to-real transfer methods with CAPS as follows: (a) SAC only (without CAPS, without sim-to-real transfer method). (b) SAC + CAPS. (c) SAC + DR. (d) SAC + DR + CAPS. (e) SAC + CycleGAN. (f) SAC + CycleGAN + CAPS.

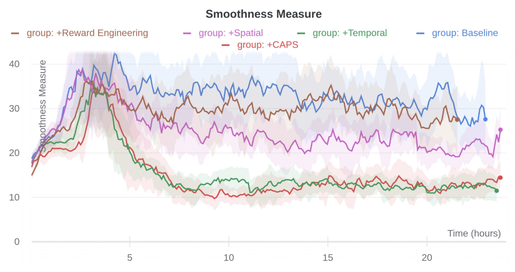
For the domain randomization methods, we apply three types: Salt and Pepper noise, Random Reflection, Random HSV Shift. The agent is trained in the simulation with the sim-to-real transfer methods and then tested in the real track 15 times with three different speed settings. In the real-world testing, we want to analyze the ability to stabilize the behaviour of CAPS by increasing the speed of the car. There is no doubt that running at a higher speed is more difficult for the agent to make proper decisions. The speed setting is a range of speed values that the car can take. We setup three different speed settings as follow: (a) *Slow*: from 1.125 to 8 (m/s). (b) *Medium*: from 1.125 to 9 m/s. (c) *Fast*: from 3.15 to 9.3 (m/s)

Table 1 and Table 2 show the completion rate and average finish lap time of all agents when testing in the real-world environment. Without both CAPS and sim-to-real transfer method, the agents cannot transfer the policies from the simulation to the real environment. The agents used domain randomization method cannot finish a track without CAPS; while CycleGAN helps the agent complete tracks in both cases (with and without CAPS). Moreover, with image-based CAPS, the behaviors of the agents are more stable, therefore can finish a run faster than the agents without CAPS. Especially, in the model that used CycleGAN as the sim-to-real transfer method, CAPS helps to reduce finish lap time from 22.20s to 17.36s when running at *slow* speed setting, and from 19.77s to 16.05s for *medium* and from 17.20s to 14.71s for *fast*, respectively.

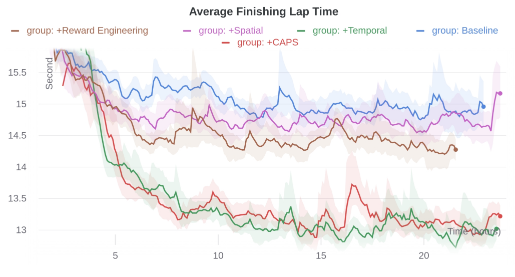
## 4.4 Study The Impact of CAPS Components

In this experiment, we study the impact of CAPS components on the objective function (Equation 1). We compare the finishing lap time and the completion rate of the following models: (a) SAC only. (b) SAC + reward engineering: Steering Angle Penalty = 0.003 * abs(degree). (c) SAC + Temporal Smoothness term. (d) SAC + Spatial Smoothness term. (e) SAC + CAPS (Spatial + Temporal).

From the comparison result in Figure 5, we can see that the Temporal Smoothness or the Spatial Smoothness individually improves the smoothness value. Moreover, the Temporal Smoothness takes the more impact on the result of CAPS when its performances are very close to the one with both Spatial and Temporal terms.

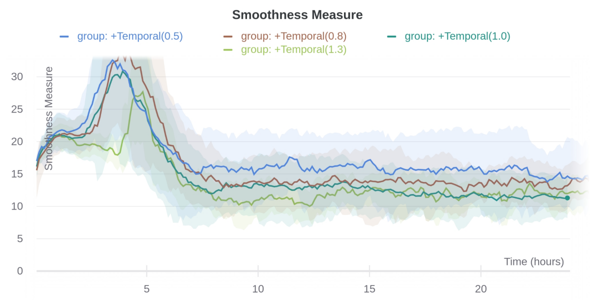


(a) Smoothness value.

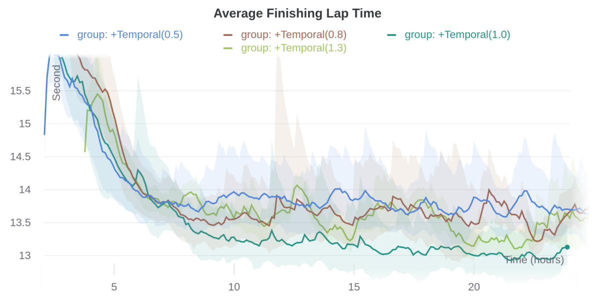


(b) Average finishing lap time.

Figure 5: Compare the impact of CAPS components.

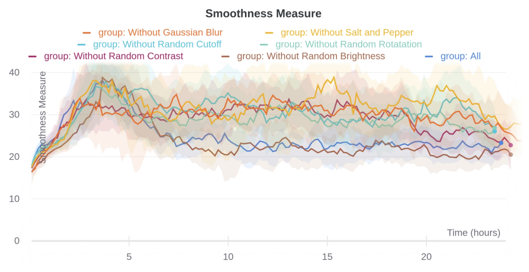


(a) Smoothness value.

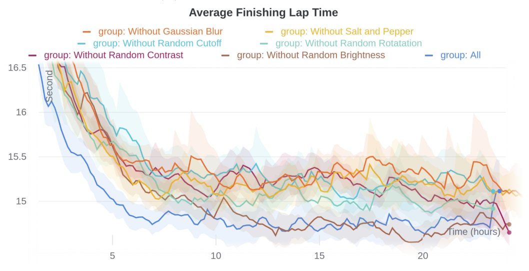


(b) Average finishing lap time.

Figure 6: Compare the sensitivity of Temporal Smoothness.



(a) Smoothness Value.



(b) Average finishing lap time.

Figure 7: Spatial Smoothness ablation study.

## 4.5 Temporal Smoothness Sensitivity Analysis

From the experiment above (4.4), we see that the Temporal Smoothness has more impact on improving the agent's performance. In this experiment, we analyze the sensitivity of the Temporal Smoothness in CAPS. We remove the Spatial Smoothness term by setting the $\lambda_S = 0$. We then vary different choices of $\lambda_T$: 0.5, 0.8, 1.0, 1.3.

Figure 6 shows that $\lambda_T = 1.0$ gives the best result for both smoothness value and finishing lap time; while increasing or decreasing $\lambda_T$ value will drop the performance of the agent.

## 4.6 Spatial Smoothness Ablation Study

To implement Spatial Smoothness, we used 6 different randomization methods, as described in section 3. In this experiment, we conduct an ablation study to investigate the influence of the randomization methods on Spatial Smoothness. We individually remove each randomization method to compare with the agent that is trained with all randomization methods.

The experiment result in Figure 7 shows that removing the Gaussian Blur or Salt and Pepper caused a significant drop in performance. Ablation of Random Brightness, on the other hand, was insignificant and the result was not much different from adding this randomization method. A possible explanation is that the image captured in the simulation has less noise than the image captured in the real world. Adding Salt and Pepper helps the agent cover this type of image. Moreover, when the car runs fast, the captured images are easily blurred; thus, training with more blurred images improves the performance of the agent.

## 5 Conclusion

This paper presents image-based CAPS, an image-based regularizing action policies method to smooth the control of autonomous miniature car racing. The model that combines image-based CAPS and CycleGAN achieved the best result in real-world testing, which reduces the average finishing lap time; while improving the completion rate. Analyzing the impact of CAPS components shows that the Temporal Smoothness term takes more impact on the performance of CAPS. An ablation study of training the Spatial Smoothness term figured out that Salt and Pepper and Gaussian Blur are two important randomization methods that influencing the result.

## References

[1] Yoshua Bengio, Frédéric Bastien, Arnaud Bergeron, Nicolas Boulanger–Lewandowski, Thomas Breuel, Youssouf Chherawala, Moustapha Cisse, Myriam Côté, Dumitru Erhan, Jeremy Eustache, Xavier Glorot, Xavier Muller, Sylvain Pannetier Lebeuf, Razvan Pascanu, Salah Rifai, François Savard, and Guillaume Sicard. Deep learners benefit more from out-of-distribution examples. 15:164–172, 11–13 Apr 2011.

[2] Chen Chen, Hongyao Tang, Jianye Hao, Wulong Liu, and Zhaopeng Meng. Addressing action oscillations through learning policy inertia, 2021.

[3] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 1861–1870. PMLR, 10–15 Jul 2018.

[4] Dan Horgan, John Quan, David Budden, Gabriel Barth-Maron, Matteo Hessel, Hado van Hasselt, and David Silver. Distributed prioritized experience replay. *CoRR*, abs/1803.00933, 2018.

[5] Jemin Hwangbo, Inkyu Sa, Roland Siegwart, and Marco Hutter. Control of a quadrotor with reinforcement learning. *IEEE Robotics and Automation Letters*, 2(4):2096–2103, oct 2017.

[6] William Koch. Flight controller synthesis via deep reinforcement learning, 2019.

[7] A. Rupam Mahmood, Dmytro Korenkevych, Gautham Vasan, William Ma, and James Bergstra. Benchmarking reinforcement learning algorithms on real-world robots, 2018.

[8] Artem Molchanov, Tao Chen, Wolfgang Hönig, James A. Preiss, Nora Ayanian, and Gaurav S. Sukhatme. Sim-to-(multi)-real: Transfer of low-level robust control policies to multiple quadrotors, 2019.

[9] Siddharth Mysore, Bassel Mabsout, Renato Mancuso, and Kate Saenko. Regularizing action policies for smooth control with reinforcement learning. 2021.

[10] Haonan Yu, Wei Xu, and Haichao Zhang. Taac: Temporally abstract actor-critic for continuous control, 2021.

[11] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A. Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks, 2017.

## APPENDIX

Appendix A has more detail about using domain randomization method in Spatial Smoothness in CAPS. Appendix B describes more about the environment setup in simulation and real-world. Appendix C specifies training settings in more detail. Appendix D explains more about the implementation of domain randomization and CycleGAN; while Appendix E shows more experiment results of image-based CAPS.

## A Domain Randomization in Spatial Smoothness of CAPS

When training the Spatial Smoothness in CAPS, to generate the similar state $s'_t$ in the Spatial Smoothness in CAPS, we implement 6 domain randomization methods:

1. Random Brightness: adjust the brightness of the image.
2. Random Contrast: adjust the degree to which light and dark colors in the image differ.
3. Random Rotation: rotate the image with a random angle.
4. Salt and Pepper: add salt and pepper noise to the image.
5. Gaussian Blur: blur an image by a Gaussian function.
6. Random Cut-off: random cut-off the image by overlaying a black rectangle at a random position with random size on the top of the image.

Figure 8 demonstrates our domain randomization methods that we used to generate the similar states $s'_t$ from states $s_t$.

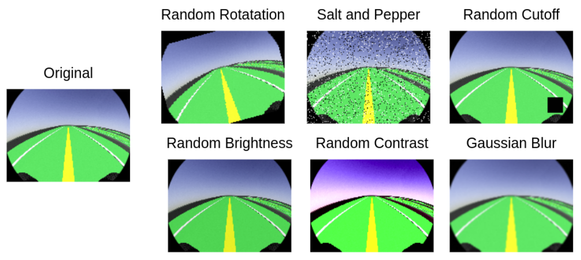
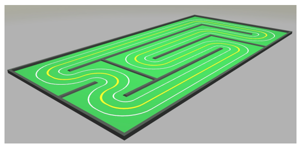


Figure 8: Domain randomization methods.

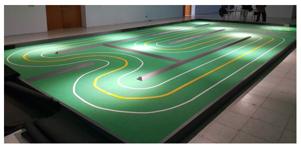
## B Environments Setup

In the real world, our track had two straight acceleration zones, two square corners, three hairpin corners, and one s-curve. If the car crosses the trapezoidal wall, it fails. In the simulation, there are no reflections, noises, or car shake on the simulator track, while the real track has distinct differences because of the dynamic sunlight changing.Figure 9 shows our simulation track and real track.

The autonomous miniature car is set up with a camera placed on the top of the car. See Figure 10. The camera captures RGB images with a resolution of 120x160 at 30fps. The computing device used in the real car is NVIDIA Jetson Xavier NX.



(a) Simulation track.



(b) Real track.

Figure 9: Simulation track(left) and real track(right).

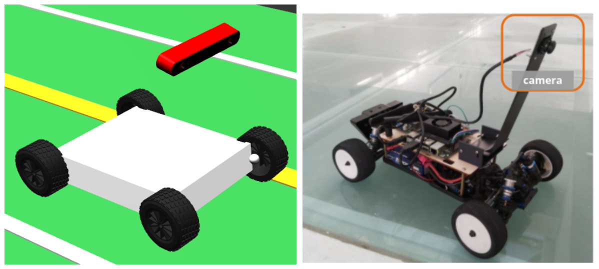


Figure 10: Simulation car(left) and real car(right).

## C Network Structure and Training

We implement a version of Deep Neural Network Ape-X[4] with Soft-Actor-Critic (SAC)[3] with the following configurations: number of workers is 3; N-step is 4; gamma is 0.98; initial alpha is 0.3; batch size is 512; learning rate is 0.0003; global buffer size is 45000; local buffer size is 2000.

The network's input is the images captured from the camera placed on the top of the miniature racing car, Figure 10. The observation is an RGB image with a resolution of 120x160, and the value of each pixel ranges from 0 to 255. We stack the current observation with the previous observation as the input. Therefore the input size is 120x160x6. The output are the continuous values of steering angle and speed with a range in [-1, 1]. Figure 11 illustrate our network structure used in this paper.

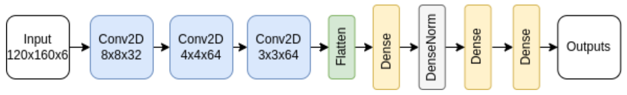


Figure 11: Network structure.

## D Domain Randomization and CycleGAN

In the experiment in subsection 4.3, we compare the effectiveness of domain randomization and domain adaptation when combined with CAPS. Here, we show more examples of the implementation of domain randomization methods and CycleGAN. For the domain randomization methods, we apply three types: Salt and Pepper noise, Random Reflection, Random HSV Shift. Figure 12 shows examples of different domain randomization methods. For the domain adaptation method, we implement CycleGAN. Figure 13 shows our CycleGAN result for translating between the simulation and real images and vice versa.

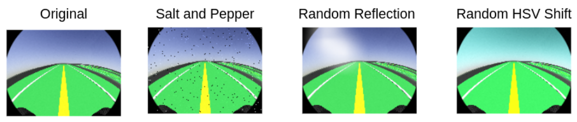
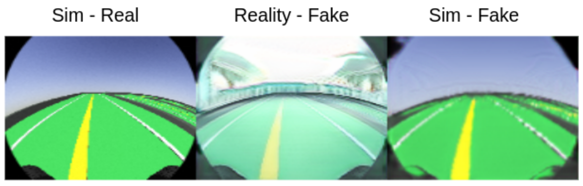
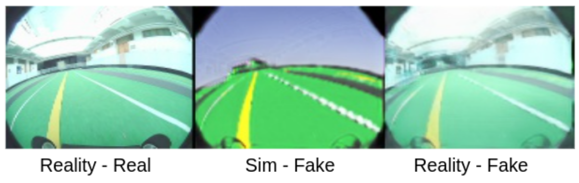


Figure 12: Domain randomization methods for sim-to-real transfer.

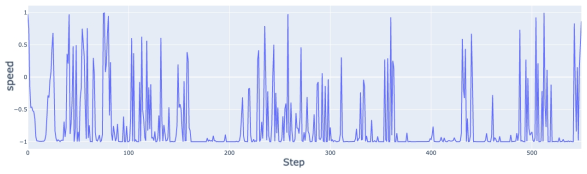


(a) From left to right: the simulation image; the generated real image by CycleGAN; the reconstructed simulation image.

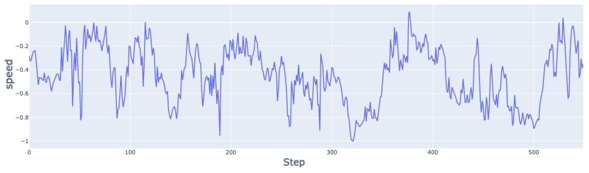


(b) From left to right: the real image; the generated sim image by CycleGAN; the reconstructed real image.

Figure 13: CycleGAN results.



(a) Speed of the car without CAPS.



(b) Speed of the car with CAPS.

Figure 14: Speed of the car with and without CAPS.

## E More Experiment Results of Image-based CAPS

In this appendix, we show more experiment results of image-based CAPS. Figure 14 is the comparison of the speed with CAPS and without CAPS. Table 3 shows the smoothness values of steering angle and speed after training model with and without CAPS. CAPS significantly improves the smoothness value in both speed and steering angle.

| | Without CAPS | With CAPS |
|---|---|---|
| Steering angle | 93.47 | **20.39** |
| Speed | 78.35 | **13.72** |

Table 3: Compare the smoothness value of steering angle and speed with CAPS and without CAPS.