Academic Journal of Science and Technology
ISSN: 2771-3032 | Vol. 13, No. 1, 2024

Developing Path Planning with Behavioral Cloning and Proximal Policy Optimization for Path-Tracking and Static Obstacle Nudging

Mingyan Zhou[1], Biao Wang[1], Tian Tan[1], Xiatao Sun[2]

[1]University of Pennsylvania, Philadelphia, PA 19104, USA
[2]Yale University, New Haven, CT 06510, USA

Abstract: In autonomous driving, end-to-end methods utilizing Imitation Learning (IL) and Reinforcement Learning (RL) are becoming more and more common. However, they do not involve explicit reasoning like classic robotics workflow and planning with horizons, resulting in strategies implicit and myopic. In this paper, we introduce a path planning method that uses Behavioral Cloning (BC) for path-tracking and Proximal Policy Optimization (PPO) for static obstacle nudging. It outputs lateral offset values to adjust the given reference waypoints and performs modified path for different controllers. Experimental results show that the algorithm can do path following that mimics the expert performance of path-tracking controllers, and avoid collision to fixed obstacles. The method makes a good attempt at planning with learning-based methods in path planning problems of autonomous driving.

Keywords: Autonomous Driving, Path Planning, Path-Tracking, Nudging.

1. Introduction

Reinforcement Learning (RL) and Imitation Learning (IL) has become more and more popular in the field of robotics. RL is a machine learning paradigm in which an agent progressively learns optimal decision-making strategies through repeated interactions with its environment. It has been improved significantly in recent years, from Vanilla Policy Gradient (VPG) [1] to most widely-used RL method, Proximal Policy Optimization (PPO) [2]. Imitation Learning (IL) involves training an agent to replicate behaviors from expert demonstrations [3] rather than learning through direct interaction with the environment. Behavioral Cloning (BC), a foundational IL method, was initially introduced by [4] using a supervised learning framework.

Beyond theoretical developments, numerous robotics applications have demonstrated the effectiveness of both RL and IL. These applications include quadrotors [5], autonomous vehicles [6], and robotic arms [7]. Several robotics platforms have been specifically developed for educational and research purposes in autonomous driving, utilizing RL and IL methods, including AutoRally [8] for off-road driving and [9] for validating RL-based algorithms. Among the various autonomous driving platforms, F1TENTH [10] stands out as one of the most promising. It features a reproducible simulation environment for rapid implementation and a wealth of open-source materials accumulated from extensive developer contributions. Significant research have been achieved using F1TENTH, including high-speed control [11], generalized RL [12], and safe overtaking [13].

In autonomous driving, classic software pipelines are modularly developed, components such as perception, planning and control [14] [15]. Besides robotics pipelines, end-to-end methods are gaining prominence in autonomous driving research due to their simplified, efficient design and potential for continuous optimization. End-to-end approaches replace either some or all of the software modules with data-driven methods, including various RL and IL techniques [16]. For end-to-end approaches developed on the F1TENTH, [17] demonstrate a successful implementation of PPO, using downsampled lidar data as input and producing steering and speed commands as output. Similarly, [18] illustrates the feasibility and the benchmark comparison of Direct Policy Learning methods. However, two challenges remain to be addressed. First, the explainability of these methods is limited, as deep learning models often function as black boxes, making it difficult to interpret their decision-making processes. This lack of transparency complicates reasoning and further validation. Second, these approaches tend to exhibit myopic behavior, as they fail to adequately account for the planning horizon, causing the vehicle to reactively respond to inputs. As observed in [19], this can lead to problematic situations, especially on right-angle tracks and in real-world environments, despite efforts to mitigate these issues through various adjustments.

To address the implicit and myopic issue, we propose the ”planning with learning” method in this study. For opaque decision-making, we replaced the planning module within the autonomous driving software pipeline similarly in [20] and [21], narrowing down the learning-based approach to particular tasks. To enhance path planning with RL and IL to move beyond reactive performance, we shifted outputs from steering and speed commands to sequences of data such as lateral offsets or a series of actions. This enables a prediction for motion planning in a longer term. Additionally, based on the prior research in [18], we utilized the strong bootstrapping capabilities of IL methods for achieving fast convergence and improved performance in the static obstacle nudging. We showed that this method can mimic the expert demonstration of path-tracking controllers, and enables the controllers with static obstacle nudging features.

This work makes three main contributions:

1) We proposed a novel approach that improves planning to ”planning with learning” as an integrated component of the robotics workflow for autonomous driving;

2) We implemented the Behavioral Cloning (BC) algorithm

on path planning in path-tracking tasks, ensuring compatibility with various path-tracking controllers (Fig. 1);

3) We utilized the Proximal Policy Optimization (PPO) algorithm bootstrapped by BC to adjust reference waypoints with lateral offsets, enabling obstacle avoidance through nudging (Fig. 2).

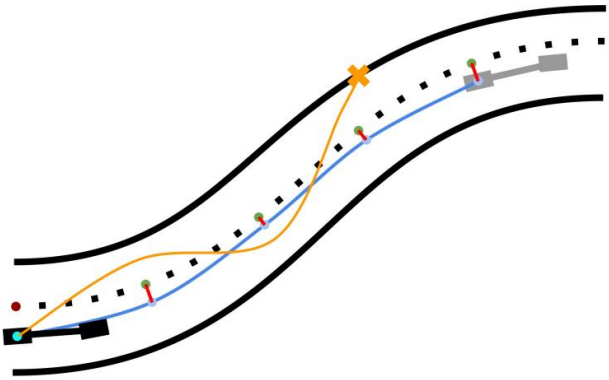


Figure 1. Tracking path through path-planning-based BC.

Given demonstration (blue trace) from the expert (single-track model in grey), instead of deviation or collision (yellow marks), the vehicle (single-track model in black) learns to mimic the expert by adjusting lateral offsets (red line segments) on the selected path (green dots) obtained by current state (cyan dot), reference waypoints (black dots), and closest waypoint (crimson dot).

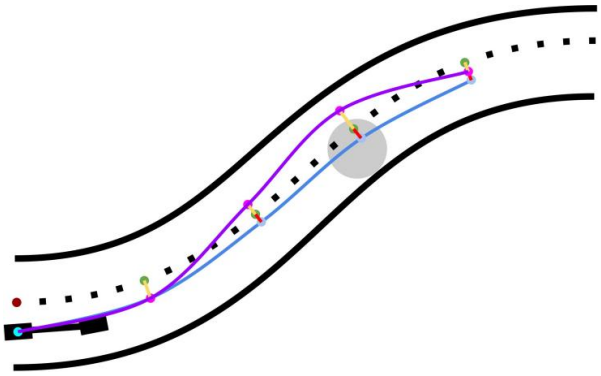


Figure 2. Static obstacle nudging through path-planning-based PPO.

After bootstrapping by BC as illustrated in Fig. 1, the vehicle performs planning similar to the expert's (blue trace). To avoid obstacles (grey circle) that may block the path, the vehicle adopts PPO to adjust the policy that outputs offsets to get a new path (purple trace), which reflects as adding new deviations (yellow line segments) to get new waypoints (pink dots).

2. Methodology

Following the classic robotics workflow, we developed processes for path-tracking and static obstacle nudging, as depicted in Fig. 3 and Fig. 4. The localization module provided the current state of the vehicle, which was then used as input for other modules, while the perception module produced lidar scans for sensing. The reference waypoints or raceline, was pre-generated as offline planning data to define the reference path. Behavioral Cloning (BC) and Proximal Policy Optimization (PPO) were employed within a "planning with learning" module, which processed three types of input data to generate lateral offsets for modifying the reference path. Finally, the controller utilized the modified path to compute steering and speed commands, allowing the vehicle to perform the desired maneuvers.

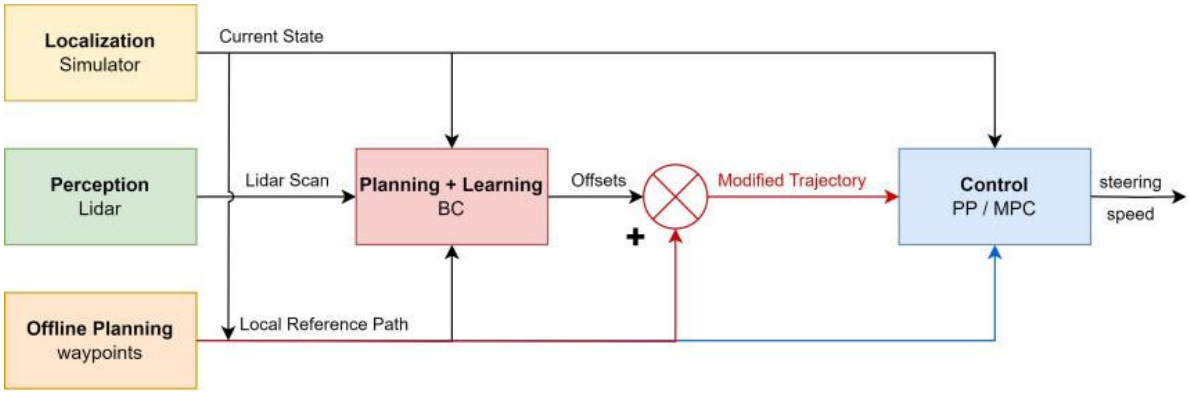


Figure 3. Structure of path-tracking with path-planning-based BC.

Controller directly takes the path from waypoints as reference (blue lines) to train the policy. During validation process, offsets are added up the to get the modified path for controller (red lines).

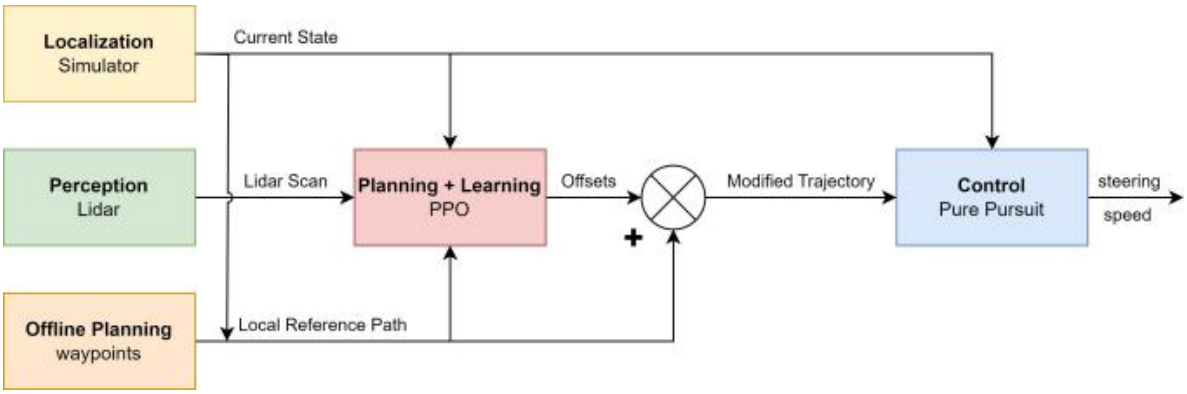


Figure 4. Structure of static obstacle nudging with path-planning-based PPO.

Bootstrapped policy by BC is trained and tested using PPO to output lateral offsets for modifying paths, thereby avoiding obstacles.

A. Kinematic Bicycle Model

Following the standard formalism of vehicle modeling, we simplified the Ackermann-steered vehicle as a single- track kinematic model [22]. In this model, the center of the rear axle represented the vehicle's position in 2D coordinates (x, y), steering angle δ, heading angle or orientation θ, and wheelbase distance L_{wb}. Suppose no side slip, we denoted the car's longitudinal velocity as v. Therefore, we defined the vehicle state $\mathbf{s}$ and desired action $\mathbf{a}$ as follows:

$$\mathbf{s} = [x, y, v, \theta], \ \mathbf{a} = [\delta_{des}, v_{des}]$$

B. Lidar Scan

The lidar scan data is defined as $\mathbf{S}_p \in \mathbf{R}^k$, where k denotes the number of lidar beams projected in the polar coordinates. To express the data more explicitly, we transformed $\mathbf{S}_p$ into Cartesian frame of the vehicle:

$$\mathbf{S} = T_p^c \mathbf{S}_p \quad (1)$$

$\mathbf{S}$ is the lidar data in local Cartesian frame, and T_p^c indicates the frame transformation.

C. Waypoints

Waypoint data is defined as $\mathrm{W} \in (\mathrm{R}^5)^{\mathrm{n}}$, where

$$w_i = [x_i, y_i, u_i, \theta_i, \gamma_i], \ i = 1, ..., n$$

denotes the coordinates, the reference longitudinal speed, the heading angle, and the curvature of the i th waypoint in n waypoints. Following waypoints as a reference path, the vehicle proceeds tracking.

In order to generate a global raceline for better reference, we applied an algorithm proposed in [23] as

$$\min_{[\alpha_1 ... \alpha_n]} \sum_{i=1}^{n} \gamma_i^2(t) \quad (2)$$

$$\text{s.t.} \quad \alpha_i \in [\alpha_{i,\min}, \alpha_{i,\max}] \quad (3)$$

Through optimization parameter α_i that indicates the relative lateral position of track boundaries, it minimizes the squared sum of every curvature γ_i at time t of spline between every adjacent waypoints. Hence, the optimal raceline is obtained as reference waypoints.

D. Path Planning

Suppose H is the planning horizon and Δt is the step time. Based on vehicle position (x, y), we extracted the waypoint coordinates the car will go in $H \cdot \Delta t$ time with H steps as $(x, y)_i$, …, $(x, y)_j$. We interpolated these waypoints and evenly sample H points including endpoints, and noted as horizon path $\mathbf{t}_h$. Through homogeneous transformation T_w^c, local horizon path is obtained. After adding offsets $\mathbf{o}$ generated by learning methods, we transformed the path with offset back to the world frame through T_w^c to get the modified path $\mathbf{t}_m$ in world frame:

$$\mathbf{t}_m = T_c^w (T_w^c \mathbf{t}_h + \mathbf{o}) \quad (4)$$

With the modified path $\mathbf{t}_m$, the path-tracking controllers can take the modified path as new reference path to perform the obstacle nudging.

E. BC for Path-Tracking

BC can be formulated through Supervised Learning as eq. (5a), where the difference between the learned policy π and expert demonstrations generated by the expert policy are minimized through loss function $\boldsymbol{L}$ with respect to some metric, and $\hat{\pi}^*$ is the approximated policy.

$$\hat{\pi}^* = \arg\min_{\pi} \sum \boldsymbol{L}(\pi(\mathbf{s}), \pi^*(\mathbf{s})) \quad (5\,\mathrm{a})$$

$$= \arg\min_{\pi} \sum_{j=1}^{t} \sum_{i=1}^{H} |o_i| \quad (5\,\mathrm{b})$$

Here, agent policy is $\pi(\mathbf{s}) = \mathbf{o}$, denotes the lateral offsets corresponding to $T_w^c \mathbf{t}_h$. Consider system dynamics and other factors, reference waypoints cannot be tracked perfectly even for the expert. However, the deviation is miscellaneous that can be ignored for simplification. L1 norm can be used as the metric to express the deviations, and the norm value in every time step can be added up as the loss function, shown as eq. (5b).

By solving this optimization problem, the policy output is trained from penalizing random sampling and large deviations to converging to the expert performance with waypoints. This can be used for bootstrapping RL methods

with rapid convergence and enhanced performance.

F. PPO for Static Obstacle Nudging

To achieve static obstacle nudging using waypoint data, lidar scans, and the current state with a bootstrapped model, we moved away from BC and other IL methods, because even expert demonstrations fall short for these tasks. Instead, we used PPO to train the policy through balancing exploration and exploitation. By focusing on policy performance without direct access to the environment, we chose policy optimization methods, specifically PPO, due to its proven performance and mature development.

In general, PPO optimizes the policy through

$$\alpha_{k+1} = \arg\max_{\alpha} E[\min(\frac{\pi_{\alpha}}{\pi_{\alpha_k}} A, g)] \quad (6)$$

where α_{k} denotes the policy parameters during iteration k, A is the advantage for the current policy π_{α_k}, and g is the clipping function. We referred to the implementation of CleanRL [24] with further details.

Optimizing the policy through PPO, instead of generating steering and speed commands like [18], the policy outputs o, which deviate $\mathbf{t}_h$ to be the modified path $\mathbf{t}_m$. $\mathbf{t}_m$ are then executed by Pure Pursuit, a path-tracking method, thus achieving fixed obstacle nudging.

G. Path-tracking Controllers

Pure-Pursuit To pursuit the goal, a lookahead point is determined from a fixed lookahead distance L towards the desired path. Based on the kinematic bicycle model, the geometric relationship between δ and the turning radius r ,and the arc curvature γ can be derived as:

$$\gamma = \frac{1}{r} = \frac{\tan\delta}{L_{wb}} = \frac{2|e|}{L^2} \quad (7)$$

where $|e|$ is the cross-track error from the vehicle to the lookahead point, which coordinates are provided by the reference path. Therefore, δ can be solved to achieve tracking waypoints.

Model Predictive Control By building up a optimization problem with physical constraints and state dynamics, MPC can solve for a sequence of action. Define the state z and the input u based on the single-track kinematic model as

$$z = [x \;\; y \;\; v \;\; \theta]^T, \; u = [a \;\; \delta]^T$$

where a is the desired vehicle acceleration. By discretization and linearization, system dynamics is derived, and objective along constraints are formulated as

$$\min \quad \sum_{t=0}^{H-1}(z_t - z_t^r)^T Q_t (z_t - z_t^r) + (z_H - z_H^r)^T Q_H (z_H - z_H^r) + \sum_{t=0}^{H-1}(u_t - u_t^r)^T R_t (u_t - u_t^r) + \sum_{t=0}^{H} u_t^T R_d u_t \quad (8)$$

$$\begin{aligned} \text{s.t.} \quad & z_{t+1} = Ax_t + Bu_t + C && (9a) \\ & z_0 = z_{cur}, z_{\min} \le z_t \le z_{\max} && (9b, 9c) \\ & u_{\min} \le u_t \le u_{\max}, u'_{\min} \le u'_t \le u'_{\max} && (9d, 9e) \end{aligned}$$

In (8), Q_t, Q_H stand for step and final penalty matrix of z, R_t, R_d are step and differential penalty matrix of u. (9a) indicates the system dynamics with system matrices A, B, C [25]. (9b) requires the initial condition is the current state z_{cur}, (9c, 9d) limit z_t, u_t respectively, and (9e) constrains the difference of u_t. The vehicle executes solved u_t, which is a and δ for tracking.

3. Experiments

In this section, we present detailed implementation and verification through experiments in simulation scenarios. We demonstrate that the path-planning-based BC and PPO bootstrapped by BC achieve great performance in path-tracking and static obstacle nudging respectively. The implementation of code, video links, instructions, and additional resources are available at https://github.com/derekhanbaliq/Planning-with-Learning.

A. Experimental Setup

We conducted development and verification on f1tenth_gym, a simulation environment of F1TENTH based on Gym. f1tenth_gym offers a robust closed-loop simulation framework facilitating rapid implementation. To incorporate learning-based methods, we integrated the single-file PPO implementation from CleanRL [24] into f1tenth_gym, enabling the use of PPO.

The experimental setup was deployed following various considerations. Hokuyo lidar scan data is downsampled from 1080 to 108 beams, as described in [18], to reduce the model's input dimension. The path prediction time is set to 1 second for path-tracking and 2 seconds for bootstrapping to static obstacle nudging. The planning horizon was defined as $H = 10$. The vehicle's initial pose was randomized along the reference waypoints while maintaining the same orientation and avoiding collisions. For Pure Pursuit, the vehicle was configured with a fixed lookahead distance of $L = 0.8\,\mathrm{m}$ and a constant speed of 2 m/s to ensure stable performance. Correspondingly, the speed of the MPC was capped at a maximum of 2 m/s. Additionally, the control frequency was set to 10 Hz.

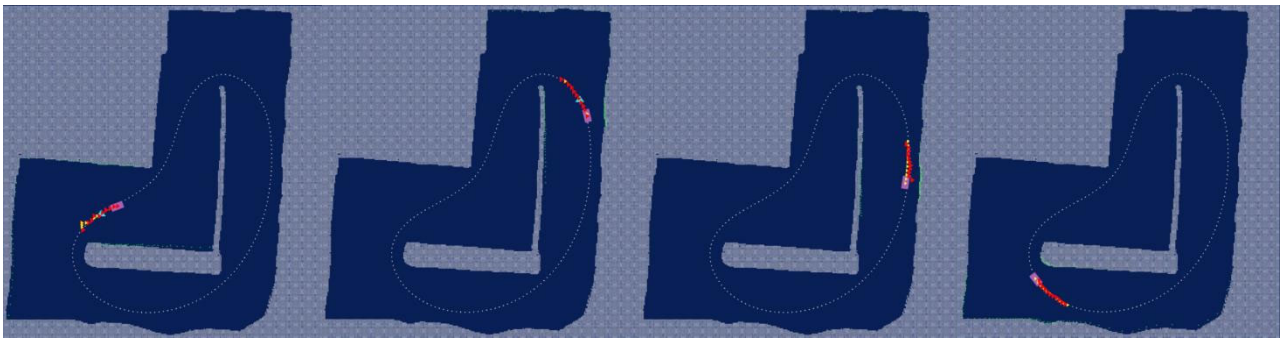
The agent model was actor-critic, compatible with the PPO design. The critic network learned the value function, which estimates the expected reward of being in a particular state; while the actor network outputted the mean of the action distribution. Both networks shared the same structure, consisting of a 4 × 256 multi-layer perceptron. The learning rate was set to 3×10−4 , the generalized advantage estimation was set to 0.95, and the discount factor was set to 0.99. The maximum gradient norm was established at 0.5 to prevent gradient explosions. To design the reward function, we calculated the reward r based on the following values: the longevity of the vehicle, which was accumulated through the number of steps n (0.01 s each) without crashing or finishing laps, the 2-norm penalty of offsets $\mathbf{0}$, and a

collision penalty $C = 1000$. The reward function was formulated as follows:

$$r = 100 \cdot n - \|\mathbf{o}\|_2 - C \tag{10}$$

B. Path-Tracking with BC

We trained agents using demonstration data that utilized Pure Pursuit and MPC controllers separately. To compare the performance of the two experiments, we set the prediction time to 1 second, consistent with the MPC setup, consistent with the MPC setup. The total number of training time steps for BC was set to 1 million, while 2 million steps were used for comparative analysis.



(a) Pure Pursuit - 1M (b) Pure Pursuit - 2M (c) MPC - 1M (d) MPC - 2M

Figure 5. The BC performance for path-tracking with Pure Pursuit and MPC using different total timesteps. Modified paths and horizon paths are marked in red and yellow. Lookahead points are shown in cyan.

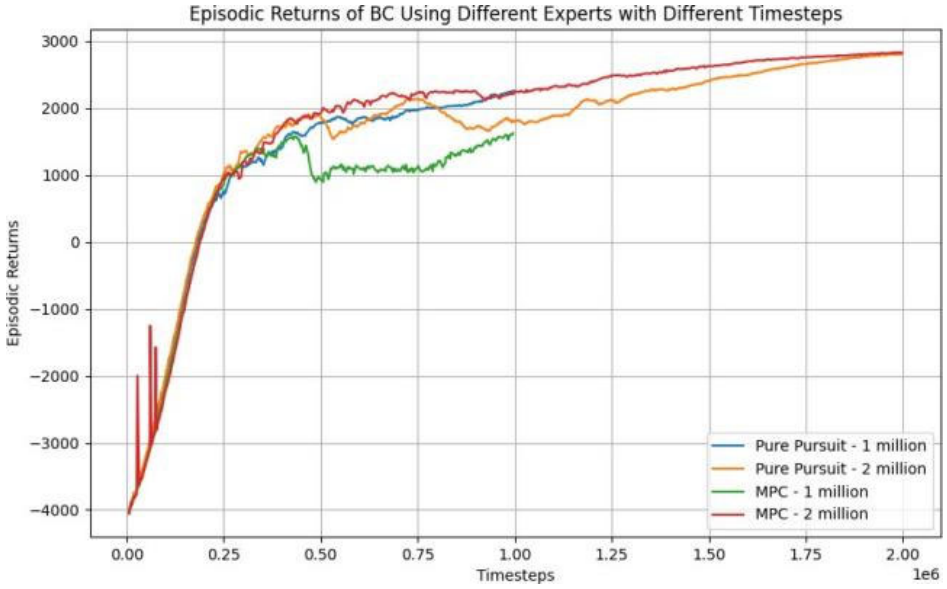
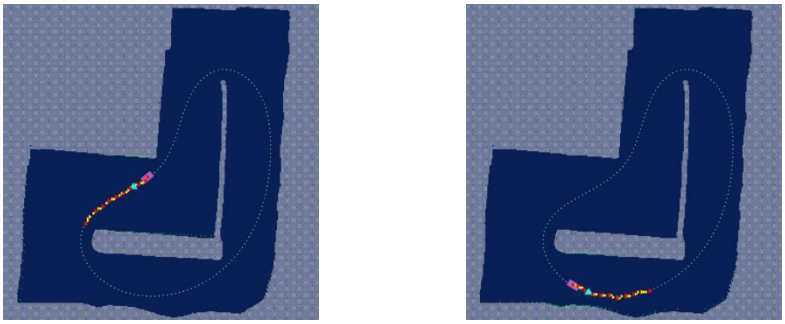


Figure 6. Episodic returns of BC for path-tracking with Pure Pursuit and MPC for different training timesteps.

For the training result, the episodic returns, shown in Fig. 6, illustrate the rapid convergence and learning efficiency of BC. With additional training, the models have higher return values than the models trained with 1 million. Besides, the 2 million models reach to nearly the same return value, indicating the method is compatible with different path-tracking controllers. Fig. 5 illustrates the modified paths are close to reference waypoints, showing the great performance of path-tracking through BC. With extended training, the paths predicted by agents become smoother with less deviation, which is also evident depicted in Fig. 6.

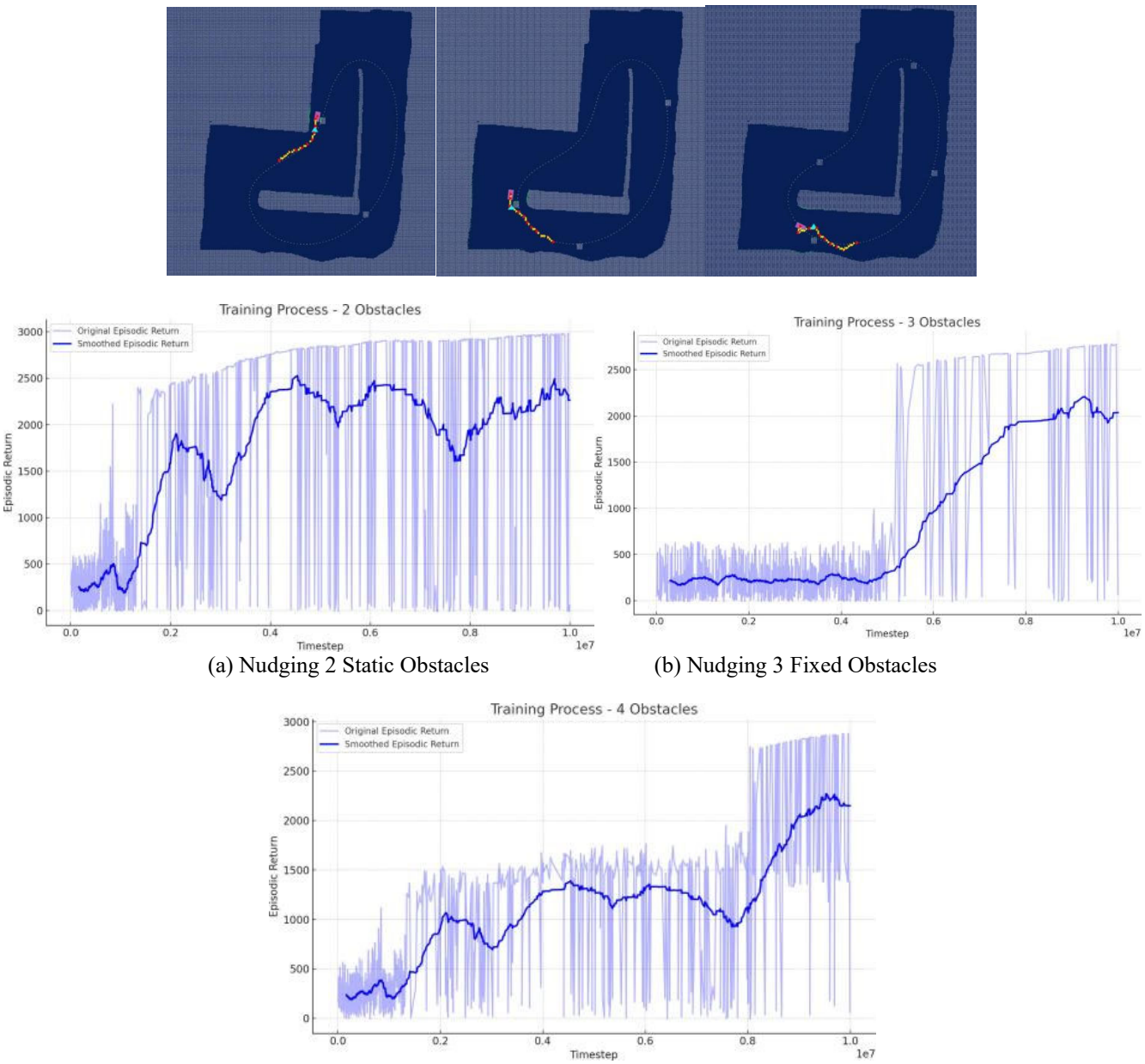


(a) 2s Prediction - 1M (b) 2s Prediction - 2M

Figure 7. The BC performance for bootstrapping static obstacle nudging with PPO in the obstacle-free map.

C. Static Obstacle Nudging with PPO

Static obstacle nudging was achieved by adjusting selected reference waypoints through lateral offsets $\mathbf{o}$. To illustrate path deviations from the horizon path, we bootstrapped models using BC with Pure Pursuit demonstrations. The models are configured with a 2 s prediction time and trained over a total of 1 and 2 million timesteps respectively. The training results are depicted as Fig. 7, which shows that both two models provide stable performance. We used 1M model for the following nudging experiments. To create an obstacle map, we modified the original obstacle-free map by adding 2, 3, and 4 static obstacles respectively. Each obstacle box measures 7 × 7 pixels, approximately 35 × 35 cm, which side length is a full width of an F1TENTH car in real scenarios. The obstacles are strategically placed along the waypoints to evaluate the vehicle's nudging performance.



(a) Nudging 2 Static Obstacles (b) Nudging 3 Fixed Obstacles

(c) Nudging 4 Fixed Obstacles

Figure 8. Static obstacle nudging and corresponding episodic returns using BC-bootstrapped PPO with Pure Pursuit validated with different fixed obstacle setups. Modified paths are marked in red triangles and yellow lines, and cyan triangles denote the lookahead points.

We used a total of 10 million timesteps to train the aforementioned model. The reward function was also modified with extra $||\mathbf{o}||_1$ term for extra penalty. The actual performances are depicted in Fig. 8. The lateral offsets $\mathbf{o}$ successfully modified the horizon path $\mathbf{t}_h$ which was truncated from reference waypoints to obtain a modified path $\mathbf{t}_m$. This modified path was then employed by Pure Pursuit, the path-tracking controller, to execute actual maneuvers. In this context, Pure Pursuit calculates a lookahead point to guide the vehicle in avoiding obstacles using $\mathbf{t}_m$. The corresponding episodic returns shown in Fig. 8 increased throughout the training process, illustrating the convergence of the policies.

4. Conclusion

In this work, we introduced a method that uses BC and PPO algorithms for path planning tasks, specifically path- tracking and static obstacle nudging. The experiments in the F1TENTH Gym environment validated that this method effectively performs both path-tracking using BC and fixed obstacle nudging using PPO bootstrapped by BC. The development demonstrated the efficacy of integrating learning into planning and highlights the practical benefits of combining RL with IL in the field of autonomous driving. Future could focus on reducing the sim-to-real gap for robust deployment, stronger generalizability to various kinds of dynamic obstacles, and decoupling the planning from decision-making to physics-constrained motion planning.

Acknowledgement

The authors express their sincere gratitude to Yi Shen from the Robotics Department at the University of Michigan for his helpful discussions and advice.

References

[1] R. S. Sutton, D. McAllester, S. Singh, and Y. Mansour, "Policy gradient methods for reinforcement learning with function approximation," Advances in neural information processing systems, vol. 12, 1999.

[2] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," arXiv preprint arXiv:1707.06347, 2017.

[3] A. Hussein, M. M. Gaber, E. Elyan, and C. Jayne, "Imitation learning: A survey of learning methods," ACM Computing Surveys (CSUR), vol. 50, no. 2, pp. 1–35, 2017.

[4] C. Sammut, Behavioral Cloning. Boston, MA: Springer US, 2010, pp. 93–97. [Online]. Available: https://doi.org/10.1007/978-0-387-30164-8-69

[5] X. Sun, Y. Wu, S. Bhattacharya, and V. Kumar, "Multi-agent exploration of an unknown sparse landmark complex via deep reinforcement learning," 2022. [Online]. Available: https://arxiv.org/abs/2209.11794

[6] H. Liu, Y. Shen, W. Zhou, Y. Zou, C. Zhou, and S. He, "Adaptive speed planning for unmanned vehicle based on deep reinforcement learning," 2024. [Online]. Available: https://arxiv.org/abs/2404.17379

[7] Zhang, K. Mo, F. Shen, X. Xu, X. Zhang, J. Yu, and C. Yu, "Self-adaptive robust motion planning for high dof robot manipulator using deep mpc," arXiv preprint arXiv:2407.12887, 2024.

[8] Y. Pan, C.A. Cheng, K. Saigol, K. Lee, X. Yan, E. Theodorou, and B. Boots, "Agile autonomous driving using end-to-end deep imitation learning," in Robotics: Science and Systems XIV. Robotics: Science and Systems Foundation, June 2018. [Online]. Available: https://doi.org/10.15607/rss.2018.xiv.056

[9] P. Cai, H. Wang, H. Huang, Y. Liu, and M. Liu, "Vision-based autonomous car racing using deep imitative reinforcement learning," IEEE Robotics and Automation Letters, pp. 1–1, 2021.

[10] M. OKelly, H. Zheng, D. Karthik, and R. Mangharam, "F1tenth: An open-source evaluation environment for continuous control and reinforcement learning," in Proceedings of the NeurIPS 2019 Competition and Demonstration Track, ser. Proceedings of Machine Learning Research, vol. 123. PMLR, 2020, pp. 77–89. [Online]. Available: http://proceedings.mlr.press/v123/o-kelly20a.html

[11] J. Becker, N. Imholz, L. Schwarzenbach, E. Ghignone, N. Baumann, and M. Magno, "Model and acceleration-based pursuit controller for high-performance autonomous racing," in 2023 IEEE International Conference on Robotics and Automation (ICRA), 2023, pp. 5276– 5283.

[12] M. Bosello, R. Tse, and G. Pau, "Train in austria, race in montecarlo: Generalized rl for cross-track f1 tenth lidar-based races," in 2022 IEEE 19th Annual Consumer Communications & Networking Conference (CCNC). IEEE, 2022, pp. 290–298.

[13] X. Sun, S. Yang, M. Zhou, K. Liu, and R. Mangharam, "Mega-dagger: Imitation learning with multiple imperfect experts," 2024. [Online]. Available: https://arxiv.org/abs/2303.00638v3

[14] J. Betz, H. Zheng, A. Liniger, U. Rosolia, P. Karle, M. Behl, V. Krovi, and R. Mangharam, "Autonomous vehicles on the edge: A survey on autonomous vehicle racing," IEEE Open Journal of Intelligent Transportation Systems, 2022.

[15] Z. Qiao, M. Zhou, Z. Zhuang, T. Agarwal, F. Jahncke, P.J. Wang, J. Friedman, H. Lai, D. Sahu, T. Nagy, M. Endler, J. Schlessman, and R. Mangharam, "Av4ev: Open-source modular autonomous electric vehicle platform for making mobility research accessible," 2024. [Online]. Available: https://arxiv.org/abs/2312.00951

[16] L. Le Mero, D. Yi, M. Dianati, and A. Mouzakitis, "A survey onimitation learning techniques for end-to-end autonomous vehicles,"IEEE Transactions on Intelligent Transportation Systems, 2022.

[17] Z. Zhuang, "f1tenth rl," https://github.com/zzjun725/f1tenth rl, 2024.

[18] X. Sun, M. Zhou, Z. Zhuang, S. Yang, J. Betz, and R. Mangharam, "A benchmark comparison of imitation learning-based control policies for autonomous racing," in 2023 IEEE Intelligent Vehicles Symposium (IV), 2023, pp. 1–5.

[19] V. Sezer and M. Gokasan, "A novel obstacle avoidance algorithm: "follow the gap method"," Robotics and Autonomous Systems, vol. 60, no. 9, pp. 1123–1134, 2012. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0921889012000838

[20] T. Weiss and M. Behl, "Deepracing: Parameterized trajectories for autonomous racing," arXiv preprint arXiv:2005.05178, 2020.

[21] L. Lipeng, L. Xu, J. Liu, H. Zhao, T. Jiang, and T. Zheng, "Prioritized experience replay-based ddqn for unmanned vehicle path planning," arXiv preprint arXiv:2406.17286, 2024.

[22] J. M. Snider et al., "Automatic steering methods for autonomous automobile path tracking," Robotics Institute, Pittsburgh, PA, Tech. Rep. CMU-RITR-09-08, 2009.

[23] A. Heilmeier, A. Wischnewski, L. Hermansdorfer, J. Betz,M. Lienkamp, and B. Lohmann, "Minimum curvature trajectory planning and control for an autonomous race car," Vehicle System Dynamics, vol. 58, no. 10, pp. 1497–1527, 2020. [Online]. Available: https://doi.org/10.1080/00423114.2019.1631455

[24] S. Huang, R. F. J. Dossa, C. Ye, J. Braga, D. Chakraborty, K. Mehta, and J. G. Arau´jo, "Cleanrl: High-quality single-file implementations of deep reinforcement learning algorithms," Journal of Machine Learning Research, vol. 23, no. 274, pp. 1–18, 2022. [Online].Available: http://jmlr.org/papers/v23/21-1342.html

[25] R. Rajamani, Vehicle Dynamics and Control. New York: Springer, 2012.