Hybrid Reinforcement Learning based controller for autonomous navigation

Ajinkya Joglekar
Automotive Engineering
Clemson University
Greenville, SC, USA
ajoglek@clemson.edu

Venkat Krovi
Automotive Engineering
Clemson University
Greenville, SC, USA
vkrovi@clemson.edu

Mark Brudnak
Ground Vehicle Systems Center
Warren, MI,USA
mark.j.brudnak.civ@mail.mil

Jonathon M. Smereka
Ground Vehicle Systems Center
Warren, MI,USA
jonathon.m.smereka.civ@mail.mil

***Abstract*—Safe operations of autonomous mobile robots in close proximity to humans, creates a need for enhanced trajectory tracking (with low tracking errors). Linear optimal control techniques such as Linear Quadratic Regulator (LQR) and Model Predictive Control (MPC) have been used successfully for low-speed applications while leveraging their model-based methodology with manageable computational demands. However, model- and parameter- uncertainties or other unmodeled non-linearities may cause poor control actions and constraint violations. Nonlinear MPC has emerged as an alternate optimal-control approach but needs to overcome real-time deployment challenges (including fast sampling time, design complexity, and limited computational resources). In recent years, the optimal control-based deployments have benefitted enormously from the ability of Deep Neural Networks (DNNs) to serve as universal function approximators. This has led to deployments in a plethora of previously inaccessible applications – but many aspects of generalizability, benchmarking, and systematic verification and validation coupled with benchmarking have emerged. This paper presents a novel approach to fusing Deep Reinforcement Learning-based (DRL) longitudinal control with a traditional PID lateral controller for autonomous navigation. Our approach follows (i) Generation of an adequate fidelity simulation scenario via a Real2Sim approach; (ii) training a DRL agent within this framework; (iii) Testing the performance and generalizability on alternate scenarios. We use an initial tuned set of the lateral PID controller gains for observing the vehicle response over a range of velocities. Then we use a DRL framework to generate policies for an optimal longitudinal controller that successfully complements the lateral PID to give the best tracking performance for the vehicle.**

Keywords—Hybrid Deep Reinforcement Learning Controller, Real2Sim, Time-Optimal Autonomous Navigation

I. INTRODUCTION

Operations with Autonomous Wheeled Mobile Robots (WMRs) in uncertain and unstructured environments have led to numerous applications ranging from on-road autonomy to automated last-mile fulfillment systems [1]. Often the requirement of conditional/full autonomy in these applications creates a need for WMRs to work in close proximity to other agents, including humans. Thus, accurate trajectory generation and tracking in dynamic environments are essential and leave very low margins for error [2][3].

Intelligent Perception-Planning-Controls behaviors now enable WMRs to make critical decisions and act without human intervention in real-time. Often, a hierarchical partitioning of realization of intelligence is pursued with: (i) an *upper-level "Perception-Planning"* stage and (ii) a *lower-level "Planning-Control"* stage. This *"Perception-Planning"* stage encompasses the process of sensing the environment (often with multiple sensor streams) and subsequent perceptual processing to enable understanding and set the stage for decision-making. In contrast, the *"Planning-Control"* stage uses controllers varying in complexity from a simple reactive PID controller to complex model-based controllers to interact with the environment. Complexities arising from unknown environments/situations now create challenges both for (i) real-time understanding and decision making, as well as for (ii) ensuing safety guarantees and controller-adaptability in uncertain environments.

Linear controllers for WMRs are attractive for providing adequate tracking capabilities with lower computational demands. Significant literature compares performance, computational efficiency, and robustness in various applications [5-8]. However, time-varying parameters, external disturbances/sensor failures, model parameters, and functional constraints could lead to poor tracking performance.[8] Approaches like Non-Linear Model Predictive Control (NMPC) can overcome these limitations but require high computational costs and sensitivity to changing system dynamics. In this setting, end-to-end data-driven machine learning-based methods have emerged as an industry approach to overcome complexity and real-time computational demands.

This is evidenced in the growth of contemporary research in autonomous driving algorithms building on AI approaches and Deep Learning-based approaches [9]. A plethora of supervised/unsupervised based AI approaches have now been deployed - ranging from subsystem deployments (like semantic segmentation, object detection, motion estimation for Perception-Planning") to end-to-end approaches (like behavior cloning for "Perception-Planning-Control") [4][9]. Reinforcement Learning approaches have found a particular niche that allows their deployment for autonomous-vehicle systems.

In this milieu, Deep neural networks have revolutionized all aspects of the discipline by their universal function approximation that allows for a deep representation learning to map high-dimensional inputs to outputs. Hence, fusing Deep Neural Networks with Reinforcement Learning gives rise to Deep Reinforcement Learning, enabling policy generation for

continuous action space problems in a model-free setup. DRL methods can provide an end-to-end workflow for various autonomy functionalities. However, these methods remain susceptible to practical challenges such as simulation-reality gap, generalizability, brittle reward functions, etc. Additionally, safety considerations must be factored in the RL decision-making process for autonomous systems. [4] [9-11]

Thus, there is a need to blend the adaptability of DRL methods with the computational simplicity and safety guarantees of a physics-guided controller realization in a DRL formulation. Such a physics-guided DRL approach can bring significant task-state-, dependency- and constraint-awareness to enhance prediction performance, generalizability, interpretability, and sample efficiency. Hence, we will use innovative approaches to integrate traditional physics-based modeling (mechanistic models, theories, and laws) with complementary strengths from state-of-the-art machine learning techniques (supervised, unsupervised, and especially in reinforcement learning settings).

To elucidate and examine these concepts in greater detail, we consider the following prototypical automotive control problem of a vehicle navigating the racetrack in an optimal trajectory. The control challenge in this scenario can be broken down into lateral and longitudinal control. A traditional handcrafted PID lateral controller permits operations within a test environment but with the goal of bootstrapping a physics-aware DRL longitudinal controller. The performance of this combined PID and DRL controller is verified against the lateral reference controller sampled at different velocities.

The paper is organized as follows. Section II presents our project's experimental setup in the F1/10th scaled vehicle platform and Software in Loop (SIL) toolchain. Section III delves into the construction of longitudinal and lateral controllers. Section IV highlights the workflow for the experimental setup. Section V presents results and performance comparisons, and we conclude with discussions on future scope and challenges in Section VI.

II. SCALED EXPERIMENTAL TESTBED

A. *F1/10th scaled vehicle platform*

Figure 1: F1/10th scaled vehicle platform [12]

Software in Loop testing and validation is a crucial part of the traditional V-model development paradigm enabling optimization of training time, rapid prototyping, and reducing negative consequences from Hardware-in-Loop testing. We build upon the F1/10th scaled vehicle ecosystem developed by researchers at the University of Pennsylvania [13] to pursue the development/deployment of autonomous driving capabilities in SIL and HIL settings. Equipped with autonomy capable hardware with an array of proprioceptive and exteroceptive sensors like LiDAR, camera, etc., it allows the vehicle to generate an intermediate representation of the environment around it, as seen in Fig. 1

B. *F1/10th simulator*

Simulator software provides a platform to set up experiments to assess the performance of our algorithms. This enables faster prototyping at a fraction of the computational and actual costs. Depending on the operating conditions, a myriad of environments can be sampled to ensure the viability/robustness of the project when transitioning to the real world. We consider various metrics for comparison noted in Santos et al. [18] for running quantitative analyses of different simulations. The default F1/10th racing simulator [13] uses a simplified bicycle dynamic model to capture vehicle performance characteristics. Within this simulator, we use three tracks of varying complexities from the F1/10th repository, as highlighted in Fig. 2 viz Torino, Berlin, and Round-Track, to test the performance of our hybrid controller.

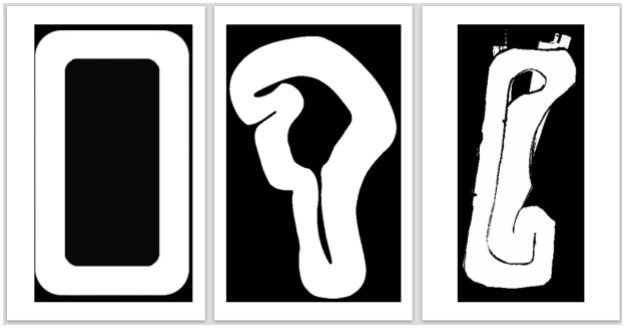

Figure 2: Test tracks on the F1/10th simulator

III. METHODOLOGY

The autonomous driving task can be on a high-level approach split up as a combination of lateral and longitudinal controllers.

A. *Lateral Bounded PID controller*

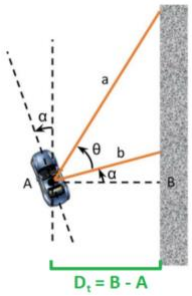

Figure 3:Wall following [12]

The goal of our reference lateral controller is to enable wall following, with a piece-wise linear wall segment, by correcting the angular offset between the wall and the vehicle. Using the

LiDAR sensor data, we can parameterize the scan to obtain both: (i) perpendicular distance, *b,* from LiDAR the o the wall; as well as (ii) the other look-ahead distance, *a,* at a fixed angle θ. Inferring from geometry, we can obtain the vehicle heading angle $\alpha = \tan^{-1}(\frac{a*\cos\theta - b}{a\sin\theta})$, which is minimized using the PID controller as demonstrated in Fig. 3,4.

$$u(t) = K_p + K_i \int_0^t e(t')dt' + K_d \frac{d(e(t))}{dt} \quad (1)$$

We use a lateral PID controller tuned using the traditional Ziegler-Nichols to minimize the track's lateral error.

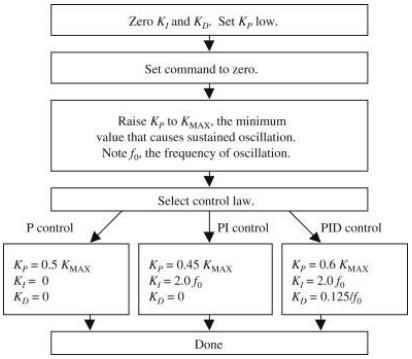

Figure 4: Ziegler Nichols method for PID tuning [16]

B. Twin Delay Deep Deterministic Policy Gradient (DDPG) based Longitudinal Controller

Reinforcement learning uses a Markov Decision Process (MDP) formulation to model a sequential decision-making process. Using the state-action-transition function-reward <*S, A, T, R*>, the RL agent develops a stochastic policy defining the probability of choosing an action given a state to attain the highest overall reward throughout the episode [4]. The Twin-Delay Deep Deterministic Policy Gradient (DDPG) algorithm is an example of the Actor-Critic method in RL, which has been widely adopted in recent years to solve reinforcement learning problems pertaining to complex continuous state-action space. It belongs to the category of off-policy actor-critic algorithms designed to overcome the limitations of the value-based method when scaled to continuous space problems [14]. As an update to the widely used DDPG for continuous state problems, TD3 improves upon the tendency of DDPG to overestimate Q-values leading to policy breaking using twin critic-networks, delaying updates to the policy networks and noise regulation to improve performance [15]. TD3 uses Clipped Q learning for smaller target values as,

$$y(r, s', d) = r + \gamma(1 - d) \min_{i=1,2} Q_{\emptyset_{i,targ}}(s', a'(s')) \quad (2)$$

and maximizes the policy through the following optimization:

$$\max_{\theta} \mathop{\mathrm{E}}_{s\sim D}[Q_{\emptyset_1}(s, \mu_\theta(s))] \quad (3)$$

$$L(\emptyset_1, D) = \mathop{\mathrm{E}}_{(s,a,r,s',d)\sim D}\left[\left((Q_{\emptyset_1}(s, a) - y(r, s', d)\right)^2\right] \quad (4)$$

$$L(\emptyset_2, D) = \mathop{\mathrm{E}}_{(s,a,r,s',d)\sim D}\left[\left((Q_{\emptyset_2}(s, a) - y(r, s', d)\right)^2\right] \quad (5)$$

The following subsections highlight the key components of our TD3 agent for optimal longitudinal control.

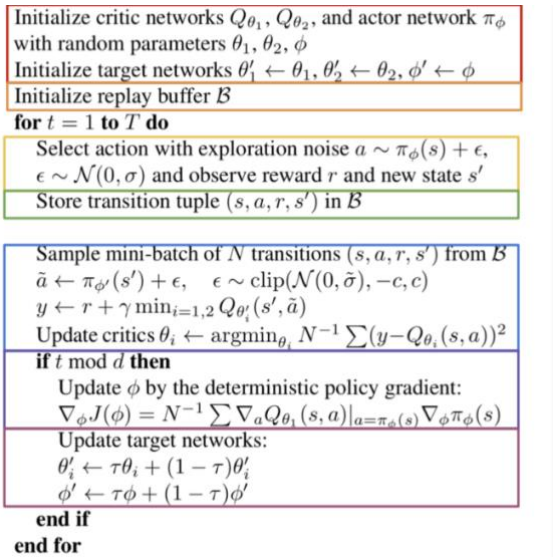

Figure 5: TD3 pseudocode [15]

Actor/Critic Networks

TD3 algorithm uses four neural networks: a deterministic actor, target actor, Q-value critic, and a target Q-value critic. The deterministic actor $\pi(s|\theta)$ returns action maximizing the long term reward while the Q-value critic $Q_k(s, a|\emptyset_k)$ returns expectation of long-term rewards due to the action from the actor. Both the target actor $\pi_t(s|\theta_t)$ and target critic $Q_{tk}(s, a|\emptyset_{tk})$ are used to improve the stability by periodically updating the actor/critic parameters. Fig. 6 shows the network diagram for our actor/critic networks.

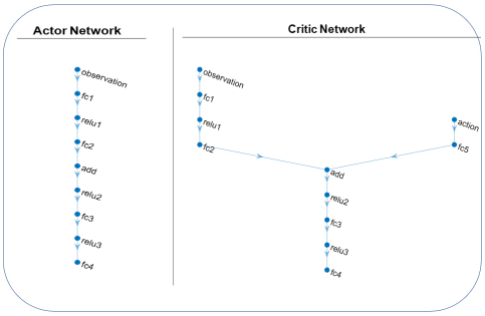

Figure 6: TD3 actor-critic networks

State Representation

At any instance of time t, the state of the system captures the interaction between the agent and the environment through observation s_t. As noted earlier, a physics-aware formulation of the RL problem can better represent the agent's interaction with its environment. Six states capture the physics of the agent in the simplified kinematic bicycle model – $[x \quad \dot{x} \quad y \quad \dot{y} \quad \varphi \quad \dot{\varphi}]$ where x, y, φ are the position and orientation of the vehicle and $\dot{x}, \dot{y}, \dot{\varphi}$ are the linear and angular velocities derived from the odometry. Additionally, the lateral error, longitudinal distance in the front using LiDAR measurement segmentation and the

previous action from the controller a_{t-1} comprise of our state observation vector.

Action Definition

Using the target policy smoothening, the TD3 actor agent generates an action based on the state which is clipped to lie in the bounded range of actions as defined in the hyperparameters for the algorithm.

$$a'(s') = clip\left(\mu_{\theta_{targ}}(s') + clip(\epsilon, -c, c), a_{Low}, a_{High}\right), \quad \epsilon \sim N(0, \sigma)$$
$$s.t \;\; a_t \in R \;\exists\; a_{min} < a_t < a_{max} \tag{6}$$

In our experiments, these limits on the forward velocity are bounded in the range [0,6.5] m/s.

Reward Function Formulation

The RL agent generates a policy that optimizes the reward function based on the problem's constraints. M. Grzes et al. [17] highlight the importance of reward function shaping on the performance of the RL agent. Considering the task in hand to generate an optimal policy for a longitudinal controller, the reward function r_t is given by:

$$r_t = \begin{cases} -\mu * error_{lateral} - \beta * (5 - a_t) \dots \\ +\gamma * distance_{total} - \delta * \dot{\varphi}, & No\ collision \\ -10, & Collision \end{cases} \tag{7}$$

The agent's goal is to maximize the reward function. There are penalties for deviating from the center of the lane, having lower velocity than 5m/s, and higher angular velocity to discourage the vehicle from going in circles. The policy rewards the total distance covered during an episode, thus balancing speed and consistent performance. The trade-off between speed and safety can be tuned by changing the value of constants $\mu, \beta, \gamma, \delta$ in the reward function

Table Column Head		
Hyperparameter	***Function***	***Value***
Sample Time	Simulation Time	0.01s
Target Smooth Factor	Parameter for target actor and critic update [0,1]	0.001
Experience Buffer Length	Number of steps of experience saved during an episode.	1e6
Discount Factor	Prioritize between immediate or overall rewards.	0.99
Mini Batch Size	Samples from the experience buffer for updating the critic model.	64

Table 1: Hyperparameter description for DRL agents

IV. IMPLEMENTATION

A. *Workflow*

Upon classifying the Reinforcement Learning problem, it is necessary to set up a workflow to interact with the simulation environment, access the required states, formulate rewards, and relay back the action from the RL policy. We used the MATLAB ROS toolbox to interface with our ROS framework and create external nodes. This capability further extends to subscribing to the network of ROS topics which encode information from the simulator in a time-synchronized method. The Deep Reinforcement Learning toolbox is then used as an abstraction to encapsulate our DRL agents. The entire workflow is illustrated in Fig. 7

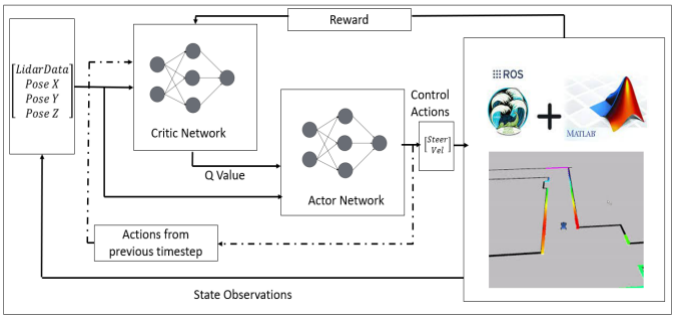

Figure 7: MATLAB ROS RL workflow

B. *Trajectories with constant linear velocity and lateral PID controller*

Using the PID controller described in Section II, we generate our reference trajectory, which is set as a benchmark to compare the performance of our DRL agent. As the lateral controller has fixed controller gains, we increase the constant longitudinal velocity and observe the vehicle response with increasing linear velocity.

C. *Training DRL agent*

Using the workflow described in Section IV.A, we train individual TD3 agents for each of the above tracks. We train the network for 1000 episodes with 1000 steps per episode. Once the DRL agent is trained, the trajectory and linear/angular velocity profiles resulting from our hybrid controller are compared and contrasted with responses from Section IV.B.

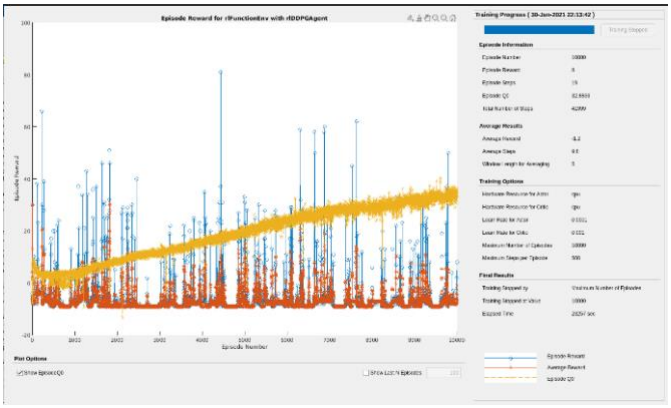

Figure 8: Snapshot of DRL agent training

V. RESULTS

A. *Comparing trajectories from DRL agent v/s PID controller*

Fig. 9,10,11 highlight the different trajectories resulting from using a lateral PID controller with varying velocities sampled at 1.5m/s,2.5m/s, and 3.5m/s (left) and resulting trajectory from the hybrid PID+RL controller (right).

Round-track

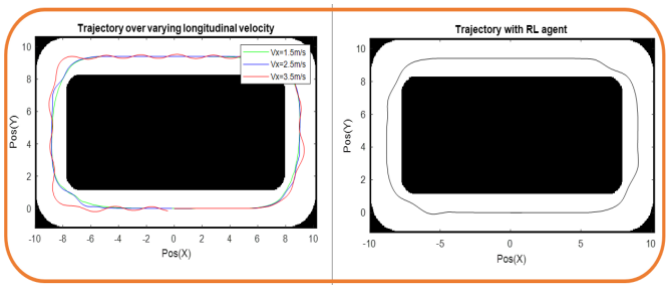

Figure 9: Round track trajectory comparison

Berlin track

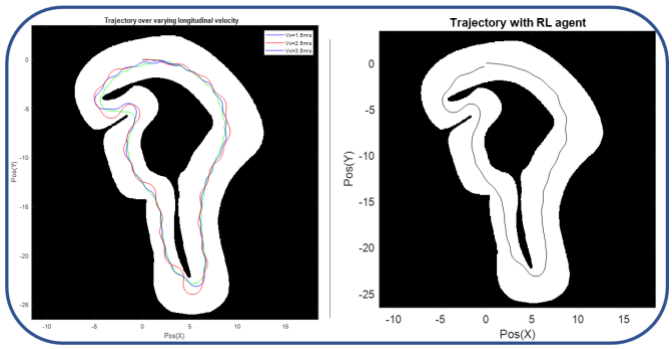

Figure 10: Berlin track trajectory comparison

Torino track

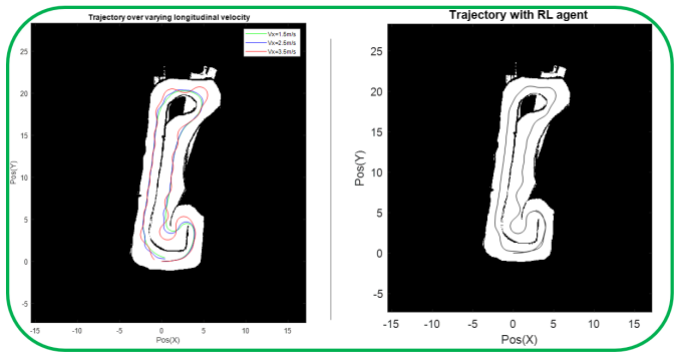

Figure 11: Torino track trajectory comparison

As expected, when the constant longitudinal velocity is increased with fixed state of tune for PID lateral controller, the vehicle starts to overshoot its desired trajectory. This leads to oscillations while the lateral controller dampens down the error. This is more prevalent in challenging circuits as shown in Fig. 10 and 11. In contrast, our hybrid "PID+DRL" controller can modulate velocity of the vehicle to maximize the rewards. This leads to increased speed during straight patches of the track and reduced speed during the corners to compliment the lateral PID controller. As a result, the oscillations are regulated considerably. This leads to smoother trajectories at higher average speed.

B. Comparative Analyses (linear and angular velcity)

Currently, we use the measured timestamped odometry data to recreate the actual angular and linear velocity profiles for the various comparative analyses. It is noteworthy that actual (average) velocities lag behind the desired input velocities, which we attribute to simulation fidelity. Fig. 12, 13 and 14 highlight the measured angular velocity profile (left) and linear velocity profile (right) as the vehicle traverses the Round, Berlin and Torino track contrasted with PID lateral controller at linear speed of 3.5m/s.

Round-track

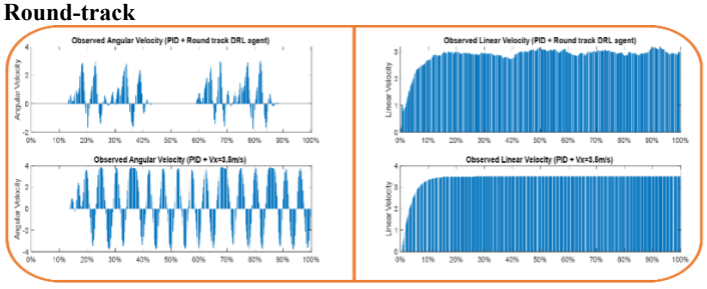

Figure 12: Round track linear and angular velocity profile

Berlin track

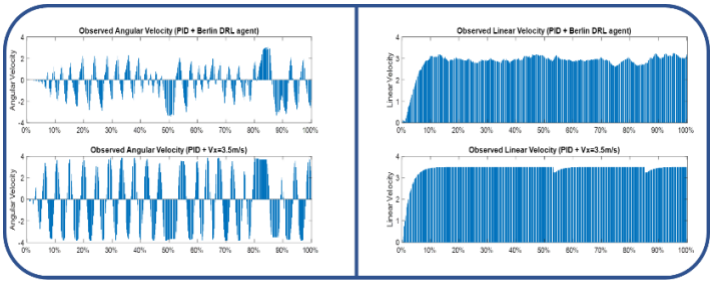

Figure 13: Berlin track linear and angular velocity profile

Torino track

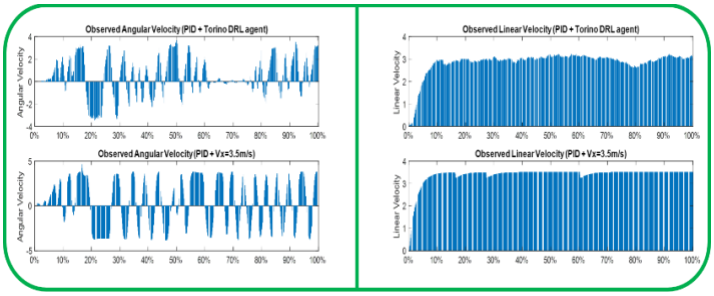

Figure 14: Torino track linear and angular velocity profile

The x-axis represents the percentage of the track covered by the vehicle over each test run. The inferences from these tests are summarized in Table II.

Metric	*PID + 1.5m/s*	*PID + 2.5m/s*	*PID + 3.5m/s*	*PID + RL*
Round-track				
Avg Lin Vel (m/s)	1.47	2.43	3.37	2.85
Avg Ang Vel (deg/sec)	0.25	0.67	2.05	0.56
Berlin track				
Avg Lin Vel (m/s)	1.48	2.455	3.39	2.83
Avg Ang Vel (deg/sec)	1.09	1.84	2.52	1.21
Torino track				
Avg Lin Vel (m/s)	1.47	2.44	3.38	2.86
Avg Ang Vel (deg/sec)	0.44	1.03	2.35	1.31

Table 2: Linear and Angular Velocity comparison

Considering the tuning of the reward function, the goal of the RL agent is to strike a balance between speed and jerk in form of angular velocity while traversing the tracks. This is evident from the average longitudinal velocity of 2.84m/s. Using interpolation via 2nd order fitting function for PID controller response at different velocities, we can compare the performance from our hybrid controller.

The hybrid PID+RL controller trained on Round-track experiences 51.1% reduction in average angular velocity compared to the response from PID controller sampled at constant velocity of 2.85 m/s. Similarly, the hybrid controller on Berlin and Torino track show 43% and 15% reduction compared to response of PID sampled at 2.83m/s, 2.86m/s respectively.

C. *Generalization to unknown tracks*

Adaptability and generalizability are potential challenges for deployment of Reinforcement Learning based applications which make a huge impact in providing safety guarantees [4]. Using a baseline lateral controller (PID) with DRL based longitudinal agent provides boundedness and generalizability. This can be measured through the performance of an RL agent in a different environment it was initially trained on. Fig. 15 highlights generalization of the trained agents across all three tracks. The 3 RL agents are the ones trained on Round-track, Berlin track and Torino track respectively. We use them in conjunction with the PID lateral controller and test performance across different tracks. The maximum fluctuation for average linear velocity (3.5%) and average angular velocity (16%) across all tracks supporting our case for performance generalization of our hybrid PID+RL based controllers on different tracks.

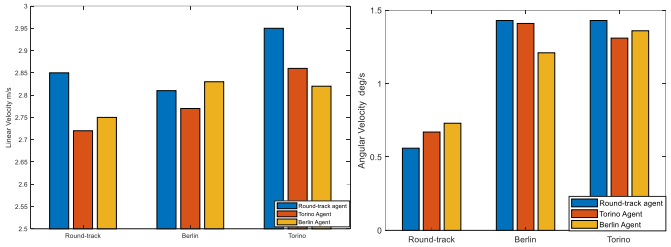

Figure 15: Average Linear and Angular Velocity comparisons

VI. DISCUSSION AND FUTURE SCOPE

In this study we examined fusing Reinforcement Learning agent with a baseline controller enables us to design a bounded and adaptable controller capable of constructing an optimal policy for the given controller gains. Consequently, the generalizability of this approach has been tested and is presented here in the form of SIL deployment.

We plan to test the validity of this SIL deployment on the real F1/10th vehicle and additional metrics based on ground truth using an indoor localization platform can be used to further improve performance. Challenges include the Sim2Real transition for RL based controllers and noisy data from the onboard sensors.

REFERENCES

[1] M. Siegel, "The sense-think-act paradigm revisited," 1st International Workshop on Robotic Sensing, 2003. ROSE' 03., 2003, pp. 5 pp.-, doi: 10.1109/ROSE.2003.1218700. *(references)*

[2] Kramer, J., Scheutz, M. Development environments for autonomous mobile robots: A survey. Auton Robot 22, 101–132 (2007). https://doi.org/10.1007/s10514-006-9013-8I. S. Jacobs and C. P. Bean, "Fine particles, thin films and exchange anisotropy," Magnetism, vol. III, G. T. Rado and H. Suhl, Eds. New York: Academic, 1963, pp. 271–350.

[3] Victerpaul, P., Saravanan, D., Janakiraman, S., & Pradeep, J. (2017). Path planning of autonomous mobile robots: A survey and comparison. Journal of Advanced Research in Dynamic and Control Systems, 9(12), 1535-65.

[4] B. R. Kiran et al., "Deep Reinforcement Learning for Autonomous Driving: A Survey," in IEEE Transactions on Intelligent Transportation Systems, doi: 10.1109/TITS.2021.3054625.

[5] Calzolari, Davide & Schurmann, Bastian & Althoff, Matthias. (2017). Comparison of trajectory tracking controllers for autonomous vehicles. 1-8. 10.1109/ITSC.2017.8317800.

[6] G. Tagne, R. Talj, and A. Charara, "Higher-order sliding mode control for lateral dynamics of autonomous vehicles, with experimental validation," in Intelligent Vehicles Symposium (IV). IEEE, 2013, pp. 678–683.

[7] M. Werling, L. Groll, and G. Bretthauer, "Invariant trajectory tracking with a full-size autonomous road vehicle," IEEE Transactions on Robotics, vol. 26, no. 4, pp. 758–765, 2010.

[8] Heß, Daniel, Matthias Althoff, and Thomas Sattel. "Comparison of trajectory tracking controllers for emergency situations." 2013 IEEE Intelligent Vehicles Symposium (IV). IEEE, 2013.

[9] Y. Ma, Z. Wang, H. Yang and L. Yang, "Artificial intelligence applications in the development of autonomous vehicles: a survey," in IEEE/CAA Journal of Automatica Sinica, vol. 7, no. 2, pp. 315-329, March 2020, doi: 10.1109/JAS.2020.1003021.

[10] P. Henderson, R. Islam, P. Bachman, J. Pineau, D. Precup, and D. Meger, "Deep reinforcement learning that matters," in Thirty-Second AAAI Conference on Artificial Intelligence, 2018. 10

[11] Y. Abeysirigoonawardena, F. Shkurti, and G. Dudek, "Generating adversarial driving scenarios in high-fidelity simulators," in 2019 IEEE International Conference on Robotics and Automation. ICRA,2019

[12] Raman, AT, Krovi, VN, & Schmid, MJA. "Empowering Graduate Engineering Students With Proficiency in Autonomy." Proceedings of the ASME 2018 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference. Volume 5A: 42nd Mechanisms and Robotics Conference. Quebec City, Quebec, Canada. August 26–29, 2018.

[13] Matthew O'Kelly, Hongrui Zheng, Achin Jain, Joseph Auckley, Kim Luong, and Rahul Mangharam, "Tech Report: TUNERCAR: A Superoptimization Toolchain for Autonomous Racing", . January 2020.

[14] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu, "Asynchronous methods for deep reinforcement learning," in Proc. Int. Conf. Mach. Learn., 2016, pp. 1928–1937.

[15] Dankwa, Stephen & Zheng, Wenfeng. (2019). Twin-Delayed DDPG: A Deep Reinforcement Learning Technique to Model a Continuous Movement of an Intelligent Robot Agent. 1-5. 10.1145/3387168.3387199.

[16] George Ellis, Control System Design Guide (Fourth Edition), 2012.

[17] M. Grzes and D. Kudenko, "Theoretical and Empirical Analysis of Reward Shaping in Reinforcement Learning," 2009 International Conference on Machine Learning and Applications, 2009, pp. 337-344, doi: 10.1109/ICMLA.2009.33.

[18] M. Santos Pessoa de Melo, J. Gomes da Silva Neto, P. Jorge Lima da Silva, J. M. X. Natario Teixeira and V. Teichrieb, "Analysis and Comparison of Robotics 3D Simulators," 2019 21st Symposium on Virtual and Augmented Reality (SVR), 2019, pp. 242-251, doi: 10.1109/SVR.2019.00049.