Available online at www.sciencedirect.com

ScienceDirect

IFAC PapersOnline CONFERENCE PAPER ARCHIVE

IFAC PapersOnLine 55-37 (2022) 400–407

Deep Reinforcement Learning Based Adaptation of Pure-Pursuit Path-Tracking Control for Skid-Steered Vehicles

Ajinkya Joglekar, [1] **Sumedh Sathe, Nicola Misurati, Srivatsan Srinivasan, Matthias J. Schmid, Venkat Krovi**

Dept. of Automotive Eng., Clemson University, Greenville, SC 29607 (e-mails: [ajoglek, nmisura, ssathe, srivats, schmidm, vkrovi]@clemson.edu).

Abstract:
The growing need for autonomous vehicles in the offroad space raises certain complexities that need to be considered more rigorously in comparison to onroad vehicle automation. Popular path control frameworks in onroad autonomy deployments such as the pure-pursuit controller use geometric and kinematic motion models to generate reference trajectories. However in the offroad settings these controllers, despite their merits (low design and computation requirements), could compute dynamically infeasible trajectories as several of the nominal assumptions made by these models don't hold true when operating in a 2.5D terrain. Outside of the notable challenges such as uncertainties and non-linearities/disturbances introduced by the unknown/unmapped 2.5D terrains, additional complexities arise from the use of vehicle architectures such as the skid-steer that experience lateral skidding for achieving simple curvilinear motion. Additionally, linear models of skid-steer vehicles often consist of high modeling uncertainty which renders traditional linear optimal and robust control techniques inadequate given their sensitivity to modeling errors. Nonlinear MPC has emerged as an upgrade, but needs to overcome real-time deployment challenges (including slow sampling time, design complexity, and limited computational resources). This provides an unique opportunity to utilize data-driven adaptive control methods in tailored application spaces to implicitly learn and hence compensate for the unmodeled aspects of the robot operation. In this study, we build an adaptive control framework called Deep Reinforcement Learning based Adaptive Pure Pursuit (DRAPP) where the base structure is that of a geometric Pure-Pursuit (PP) algorithm which is adapted through a policy learned using Deep Reinforcement Learning (DRL). An additional law that enforces a mechanism to account for the rough terrain is added to the DRL policy to prioritize smoother reference-trajectory generation (and thereby more feasible trajectories for lower-level controllers). The adaptive framework converges quickly and generates smoother references relative to a pure 2D-kinematic path tracking controller. This work includes extensive simulations and a bench marking of the DRAPP framework against Nonlinear Model Predictive Control (NMPC) that is an alternate popular choice in literature for this application.

Keywords: Adaptive control, Path tracking, Offroad systems, Pure pursuit, Deep reinforcement learning, Data-driven control

1. INTRODUCTION

The design-simplicity and physical robustness of 4-wheel skid-steer systems coupled with its high maneuverability make it very desirable for a range of military and civilian off-road applications (Khan et al., 2020). At a kinematic-level, this design simplicity is a massive advantage, however path control frameworks that utilize these models could often generate dynamically infeasible trajectories during high speed operations due to unmodelled/uncertain dynamics. Consider the exemplary case-study of a geometric Pure-Pursuit (PP) controller that is commonly used in autonomous vehicle applications. Such a path-tracking vehicle controller takes input waypoints (provided by a global path planner) and creates a trajectory reference for lower-level dynamic controllers. Despite its merits, such controllers often assume nominal operating conditions and idealistic motion models that don't hold true during highly dynamic operations in an unmapped/uncertain 2.5D terrain.

System identification and accurate modeling of the system aided by robust control laws are often used to tackle uncertainties. However, despite best efforts towards modeling the system (Aguilera-Marinovic et al., 2017; Economou et al., 2002) additional challenges arise due to factors like non-permanent wheel-ground contact, loss of traction and ground deformation. Popular modern linear optimal and robust control techniques are extremely sensitive to these modeling-, parameter uncertainties and nonlinear-

[1] Corresponding Author

10.1016/j.ifacol.2022.11.216

ities/disturbances. As a result, model-dependency is a crucial limiting factor to successful deployments in varied application spaces (on-road, off-road and manufacturing shop floor). However, systems with parameter uncertainty and complex nonlinearities in some areas of the operating domain are prime applications for adaptive control methods that can effectively cope with such issue sets (Slotine and Li, 2005; Srinivasan et al., 2021).

In this setting, end-to-end data-driven machine learning-based methods such as Deep Reinforcement Learning (DRL) have emerged as a popular approach to generate a control law that learns from real world conditions (Chen and Chan, 2020; Gheisarnejad and Khooban, 2021; Goel and Chauhan, 2021). While an end-to-end learning approach can potentially elevate path tracking control, learning decent policies in a large state-action space can increase training time and add heavy dependency on complex and sporadic reward functions. Thus, fusing learning based approaches with traditional controllers (geometric, kinematic, dynamic) lends itself to methodical formulations with the ability to provide baseline guarantees. Successful deployments of the same in planar path tracking problems have been demonstrated by Park et al. (2014); Sukhil and Behl (2021) and Joglekar et al. (2021).

In this study, we advance the geometric pure-pursuit controller by auto-tuning appropriate control parameters adaptively using the optimal policy generated by a DRL framework. We call this modified framework DRAPP (Deep Reinforcement Learning based Adaptive Pure-Pursuit) and henceforth we will refer to it as such. To demonstrate these capabilities, scenarios were built wherein a skid steer robot has to accurately follow a geometric path in 2.5D environment while traversing over perturbations in form of bumps along the way. In order to benchmark the performance of our proposed framework we choose the popular Nonlinear Model Predictive Control (NMPC) approach that has been tried and tested for this application (Kim et al., 2017; Zhao et al., 2019).The strength of this study is that despite the geometric nature of the DRAPP's explicit formulation, it possesses the ability to implicitly gather an intuitive understanding of the vehicle dynamics upon which it trains. This ability is lost on methods such as NMPC where explicit modeling of the vehicle dynamics is required if dynamics considerations need to be made.

The contributions of this study are as follows: contributions:

(1) A novel reinforcement learning based controller capable of augmenting traditional path tracking controller's performance by adapting the look ahead distance and linear velocity. Our hybrid controller implicitly captures and handle system dynamics in off-road setting via experience based learning enabling accurate path tracking.
(2) Illustration of the ability to use reinforcement learning techniques to deploy a platform-agnostic controller tailored to serve metrics that are most important for a particular application without adding monumental design, formulation and tuning complexities to the controller. This is demonstrated by the ability of DRAPP to adapt to 2.5D terrain despite bumps and perturbations by simple additions to the reward function. This is further juxtaposed with the tuning and design complexities involved in successfully deploying an NMPC controller for the same application.

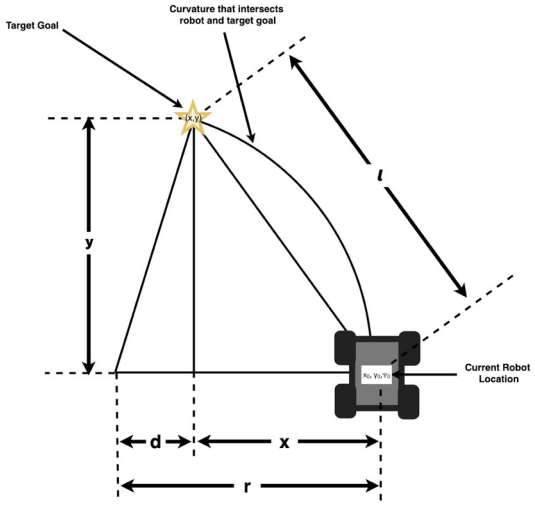

Fig. 1. Geometric tracking using pure-pursuit controller

The remainder of the paper is organized as follows: Section 2 briefly summarizes the geometric model of a skid-steered vehicle followed by the derivation of the pure-pursuit controller hence setting up the problem for our study. Section 3 presents the background, design, training and deployment approach of the employed DRL scheme. In order to setup a comparison/benchmarking metric, we describe the NMPC framework design aspects in section 4. In terms of results, section 5 introduces the simulation and software tools equipped in this study followed by section 6 where the results of the testing and validation performed are presented.

2. PURE-PURSUIT CONTROLLER

In our study, we use the seminal pure-pursuit algorithm (Coulter (1992)) as a baseline to impose a feedback law which is governed by a DRL agent

Traditional pure-pursuit: Let the waypoints that the robot needs to track be represented by

$$\mathcal{W} = \{(x_i, y_i) \quad i \in 1 \rightarrow N\} \tag{1}$$

Consider a general skid-steer drive robot, R, whose initial pose is defined in map coordinates as (x_0, y_0, ψ_0) (longitudinal position, lateral position and orientation respectively). We define l_d to be the look ahead distance parameter for the pure-pursuit controller. Let this look ahead intercept the waypoints at some (x_i, y_i). In our DRAPP controller, we consider this look ahead point at any given time by $(l_{x,t}, l_{y,t})$ to tune for the look ahead distance l_d.

Based on Fig.1, the radius of curvature to track the point (x_i, y_i) is given by

$$r = \frac{l_d^2}{2y_i}. \tag{2}$$

The robot requires two high level trajectory references namely the linear and angular velocity (v, ω). Given a cer-

tain linear velocity and radius of curvature (as computed in Eq.(2)), angular velocity is determined using $\omega = v/r$.

3. DEEP REINFORCEMENT LEARNING FRAMEWORK

3.1 Deep Reinforcement Learning Architecture

This section highlights our implementation of a DRL agent used to generate a policy π(s,a), which tunes the look ahead distance l and velocity v_t in the pure-pursuit controller formulation. The state-action representation (s,a) of the environment described in 3.1.2 sets up the platform for experience based learning to generate an optimal policy π^* to maximize the Q-value function as highlighted in Eq. (3). This Q-value corresponds to the reward function Eq. (8), (9) which is to be maximized.

Deep Reinforcement Learning Agent: Given the nature of our problem being a high-dimensional stochastic environment with no preview, an approach that provides the the sampling efficiency of value-based DRL methods and the convergence of policy-based DRL methods is necessary. Actor-critic algorithms offer benefits of both policy-based and value-based methodologies in an adversarial setting making them desirable for our problem setup. Our DRL agent is based upon the Twin-Delay Deep Deterministic Policy Gradients (TD3) (Dankwa and Zheng, 2019). There are two key facets to the TD3 implementation: The "actor network" which uses a policy gradient to maximize the reward function $Q_\phi(s,a)$ through the expected returns E given by:

$$\max_{\theta} \underset{s\sim\mathcal{D}}{\mathrm{E}} [Q_{\phi_1}(s, \mu_\theta(s))] \tag{3}$$

The "critic network" in the TD3 algorithm uses a single target function y as seen in Eq. (4) derived from the network having the least target value of Eq. (5). In this equation d indicates whether the state s' is terminal. The parameter $\mathcal{D}$ is the replay buffer for updating the Q-networks.

$$y(r, s', d) = r + \gamma(1 - d) \min_{i=1,2} Q_{\phi_{i,\mathrm{targ}}}(s', a^{'}(s')) \tag{4}$$

$$L(\phi_i, \mathcal{D}) = (s, a, r, s', d) \sim \mathcal{D}\big(Q_{\phi_i}(s, a) - y(r, s', d)\big)^2 \tag{5}$$

where $i = 1, 2$. The TD3 algorithm learns these two functions Q_{ϕ_1} and Q_{ϕ_2} using the Mean Squared Bellman Equation as seen in Eq. (5). Extensive formulation of TD3 and nomenclature for Eq. (3) - (5) can be found in Dankwa and Zheng (2019).

RL agent state-action representation: In order to achieve better tracking performance in an offroad environment, the DRL agent not only needs to consider pose, cross-track error, prior look ahead point $(l_{x,t}, l_{y,t})$ and velocity (v_t) but also the pitch velocity observed by the on-board IMU as the vehicle traverses over the the 2.5D environment(ω_y). Thus, the state representation is a [7x1] dimensional vector given by,

$$s_t = [x_{0,t}, y_{0,t}, \psi_{0,t}, \omega_y, l_{x,t}, l_{y,t}, v_t]' \tag{6}$$

The output from the DRL agent is the tuneable look ahead parameter and the linear velocity to the skid-steer robot.

$$a_t = [l_d, v_t] \tag{7}$$

where $l_d \in [0.2m, 1.5m]$ and $v_t \in [0.1m/s, 2m/s]$.

3.2 Stability Considerations:

Optimization function: We frame two reward functions for training two different RL agents, one without considering the angular velocity observed by IMU as the vehicle travels over bumps in the environment Eq.(8) and the other penalizes traversing over bumps at high speeds to avoid destabilizing the robot and affecting tracking performance Eq. (9).

$$R_{t1} = \alpha_1\|\epsilon_{ct}\|^2 + \alpha_2(v_{ref} - v_t)^2 + \alpha_4(d_t) \tag{8}$$

$$R_{t2} = \alpha_1\|\epsilon_{ct}\|^2 + \alpha_2(v_{ref} - v_t)^2 + \alpha_3(\omega_y)^2 + \alpha_4(d_t) \tag{9}$$

The parameters in Eq. (8)(9) are cross-track error$(\|\epsilon_{ct}\|)$, target velocity (v_{ref}), total distance covered by the robot (d_t) and rotational velocity in ω_y measured by the on-board IMU. $\alpha_1, \alpha_2, \alpha_3, \alpha_4$ are the weights assigned to the penalties.

Hyper parameters and computational time: We consider the following hyper parameters for training the DRL agent.

Table 1. Experiment hyper parameters

Hyperparameters	
Sample Time	0.01
Training episodes	50
Training steps per episode	1500
Cross track threshold (for failure)	0.8
Distance to goal threshold	0.2
Mini batch size	64
Experience buffer length ($\mathcal{D}$)	1e6
Discount Factor (γ)	0.99

Given the hyper parameters, training the network typically takes about 3 hours for a single policy. However, once the network is trained for the ideal policy, the computational time required for one forward pass is $\sim 60ms$.

4. NONLINEAR MODEL PREDICTIVE CONTROL

In order to benchmark the performance of the proposed DRAPP controller, a Nonlinear MPC is deployed in a path tracking capacity. The dynamic equations of motion of skid-steered systems are considerably complex and expensive to utilize for real-time motion control (Mandow et al. (2007)). Hence, kinematic model predictive controllers are preferable under low-speed driving conditions due to their simplicity and computational efficiency (Tang et al. (2020)). Given the higher-level path tracking nature of the objective and low operating speeds,a kinematic NMPC has been utilized.

The kinematic model used in our study is an extended differential drive model summarized by Rabiee and Biswas (2019) through the equations,

$$v = (\omega_l + \omega_r)\frac{r_w}{2} \tag{10}$$

$$\omega = (-\omega_l + \omega_r)\frac{r_w}{w\chi} \tag{11}$$

where ω_l and ω_r are the left and right wheel angular velocities, respectively, and v and ω are longitudinal and

angular velocities of the robot respectively. The effective wheel radius is denoted by r_w, w is the lateral track width, and χ is an empirical parameter dependent on location of instantaneous center of rotation. It assumes values $[1, \infty)$, $\chi = 1$ being the case corresponding to an ideal differential drive robot.

The primary objective of the NMPC is to track a predetermined path as closely as possible. This is accomplished by penalizing the position error with respect to the waypoints ($\epsilon = [x_r - x, y_r - y]^T$), where (x_r, y_r) is the position of the robot. In order to result in a smooth input profile that prevents sharp accelerations in the vehicle, we also penalized the rate of change of control input $\mathbf{u(t)}$. Additionally, the proposed NMPC should be able to track a reference $\mathbf{u_{ref}(t)}$. The control input $\mathbf{u(t)} = [\omega_l(t), \omega_r(t)]^T$ is comprised of wheel velocities described in Eq. (10) (11)

The kinematic behavior described in Eq. (10)(11), is represented by equality constraint in Eq. (14b), in addition to actuator limits, feasible state spaces and initial constraint.

The state of the vehicle is represented by $\mathbf{q(t)} = [x_r(t), y_r(t), \psi(t)]^T$. The optimal control problem is solved over a fixed finite horizon N_p, subscribing to the Euler-forward integration scheme shown below.

$$\dot{\mathbf{q}}(\mathbf{t+1}) = \mathbf{f}(\mathbf{q(t)}, \mathbf{u(t)}) = \begin{bmatrix} (\omega_l(t) + \omega_r(t)).\frac{r_w}{2}.\cos\psi(t) \\ (\omega_l(t) + \omega_r(t)).\frac{r_w}{2}.\sin\psi(t) \\ (\omega_r(t) - \omega_l(t)).\frac{r_w}{w\chi} \end{bmatrix} \tag{12}$$

$$\mathbf{q(t+1)} = \mathbf{q(t)} + \mathbf{f}(\mathbf{q(t)}, \mathbf{u(t)}).\Delta t \tag{13}$$

where Δt is the time step of integration.

$$\min_{u} \sum_{t=1}^{t=N_p} \epsilon^T Q \epsilon + (\frac{\mathbf{u(t)} - \mathbf{u(t-1)}}{\Delta t})^T P (\frac{\mathbf{u(t)} - \mathbf{u(t-1)}}{\Delta t}) + (\mathbf{u(t)} - \mathbf{u_{ref}(t)})^T R (\mathbf{u(t)} - \mathbf{u_{ref}(t)}) \tag{14a}$$

subject to

$$\mathbf{q(t+1)} = \mathbf{g}(\mathbf{q(t)}, \mathbf{u(t)}) \quad \forall t \tag{14b}$$

$$\mathbf{u_{min}} \leq \mathbf{u(t)} \leq \mathbf{u_{max}} \ \forall t \tag{14c}$$

$$\mathbf{q_{min}} \leq \mathbf{q(t)} \leq \mathbf{q_{max}} \ \forall t \tag{14d}$$

$$\mathbf{q(0)} = [x_0, y_0, \psi_0]^T \tag{14e}$$

In NMPCs, the tunable parameters -penalty matrices Q, P and R, prediction horizon length N_p, and control horizon length N_c, have complex relationships with the optimization variables. Due to the lack of methodical tuning approaches for MPC controllers, as described by Shah and Engell (2011), the NMPC controller was tuned and tested against various conditions enlisted below.

(1) **Maximum Tracking Performance :** In order to satisfy this condition, the tunable weight R, was set to zero to let the controller freely select control input $\mathbf{u(t)}$ within the specified range. Consequently, the NMPC is biased towards minimizing ϵ.
(2) **Minimum Control Effort :** To meet this condition, the reference $\mathbf{u_{ref}(t)}$ was set to zero. The parameters Q and R were tuned to ensure that minimum cumulative control effort is spent while ϵ remains bounded.
(3) **Maximum Speed :** This condition is satisfied by setting the reference $\mathbf{u_{ref}(t)}$ equal to the maximum control input $\mathbf{u_{max}}$.

The prediction horizon N_p was tuned for best performance and $N_p = 1$ second was determined to be optimal preview. Similarly $N_c = 0.1$ seconds was selected given the fast moving nature of this agent. For comparison with DRAPP, the weight matrices were set as $Q = 30$, $R = 0.1$, $P = 0.1$.

The Interior Point Optimizer Solver (IPOPT) (Wächter and Biegler, 2006), implemented through the CasADi nonlinear optimization tool developed by (Andersson et al., 2019), was used to solve the NMPC optimization.

5. TOOLS OF STUDY

This section presents the tools that were used to perform the deployment aspects of this study. This includes the simulation platform and the environment and process pipeline for training our DRAPP algorithm.

5.1 Gazebo Environment

To represent our environment with varying surface geometry, we have used the ROS Gazebo simulation engine. The process of generating such an environment can be described as:

(1) **Environment and robot**: We create our desired environment using the Gazebo building editor which enables us to add obstacles from CAD files. The *.urdf* file for the Clearpath Husky (Clearpath, 2020) is used to spawn an exemplar robot in the said environment.
(2) **Mapping and Localization**: Since accurate state-estimation is outside the scope of this study, to obtain consistent pose information, we map out the environment using an off-the-shelf Hector SLAM package and localize the robot in the map frame to obtain our pose estimate.

5.2 Reinforcement Learning Pipeline

We use the pure-pursuit algorithm with our DRL agent to learn the policies for our DRAPP controller.
We subscribe to sensor data from the simulation environment, feed in the input observations to the DRAPP agent, and obtain actions from the controller. Once the control action from the RL agent is implemented on the robot, we pause the simulation to get the observations used for calculating rewards and to update the weights of our actor-critic networks. This whole process is repeated until the termination condition for a training episode is triggered, after which the robot is reset in the environment.

6. RESULTS

In this section, we compare and contrast performance of three different approaches to the path tracking problem in a uneven environment.This is highlighted using cross track

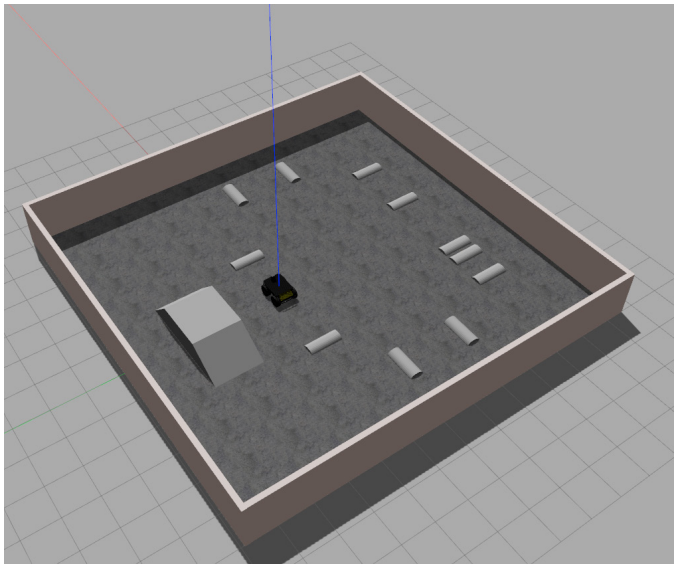

Fig. 2. Husky gazebo uneven bump environment

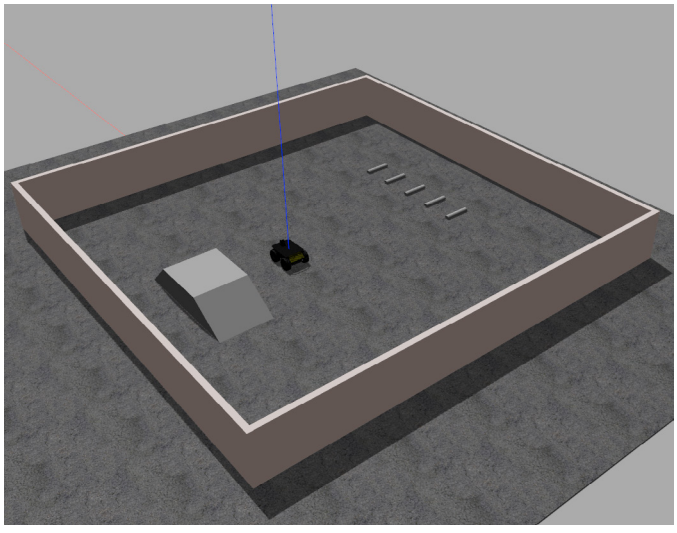

Fig. 3. Husky gazebo rumble strip environment

error (ϵ_{ct}), velocity profile (v_x) and observed pitch velocity over the bumps (ω_y) obtained from the onboard IMU sensor as the robot traverses the path. The reference path of the robot is chosen to be a rectangular planar path with intermittent bumps to emulate an uneven terrain. Having a C_0 continuous curve makes it challenging for the standard pure-pursuit with fixed look ahead distance to closely track the path. This is where we show the improvement in performance through using DRAPP controller which modulates velocity and look ahead distance to closely follow the path while considering the undulations the terrain. The performance of our controller is then validated against the NMPC for the highlighted metrics. It is to be noted that the initial spawn location of our robot is at (x_0, y_0). The performance of all three path tracking controllers viz. pure-pursuit, NMPC, and DRAPP following the rectangular path on the rumble strip environment is highlighted in Fig. 6. It is evident that both NMPC and DRAPP heavily outperform the original pure-pursuit for tracking performance as seen in Fig. 6. Further increasing the speed on the pure-pursuit controller would lead to higher overshoot and sub-optimal tracking while modifying the look ahead distance would lead to undercutting or oscillations in the tracking. Considering these factors, we only compare and contrast the performance of NMPC and DRAPP from here on out for clarity.

Fig. 4, 5 compare the vehicle response with NMPC controller and two different DRAPP agents trained with Eq. (8),(9) respectively for the rumble strip environment. Both the DRAPP agents show similar performance gains in tracking and average linear velocity when compared to the NMPC deployment. The NMPC controller has a smoother velocity response just through consideration of the spatial points of the track but with no terrain information. Additionally, having smaller undulations do not perturb the robot to have significant impact on tracking.

In contrast, the uneven bump environment as shown in Fig. 2 heavily affects the NMPC controller as it does not factor into the terrain interactions for path planning, leading to sub optimal tracking compared to DRAPP agent as seen in Fig. 9. Although the NMPC formulation can be improved in order to accommodate acceleration and erk considerations that can improve this performance, this requires expertise in development as well as tuning whereas the DRAPP formulations provides an easy template to include the vertical conditions. Additionally, the DRAPP agent trained with Eq. (9) which factors into the IMU reading for angular velocity (ω_y) outperforms both the NMPC controller as well as the DRAPP agent that does not account the effect of terrain. This is evident from Fig. 8 where the DRAPP agent modulates the input velocity to the robot before traversing over the bumps. This leads to lower tracking error and reduced angular velocity (ω_y) as the vehicle significantly slows down while traversing undulations as can be contrasted between Fig. 7 and Fig. 8.

The DRAPP controller can extract maximum performance via experience based learning without prior preview of the terrain and effectively tune the look ahead distance parameter of the pure-pursuit controller while providing its own linear velocity commands to the robot. This results in overall higher speed with better tracking performance and the modulation of the velocity in the plots is indicative of DRAPP controller learning to adapt these parameters while going over bumps. It is observed that the pitch velocity (ω_y) larger for the DRAPP controller but it is a byproduct of higher velocity of the robot. Our reward function can be tuned to prioritize lower vertical jerks at expense of linear speed if needed as discussed in the Fig. 7.

6.1 NMPC Parameter Tuning

The parameters considered for tuning the NMPC controller to optimize tracking performance were the weight matrices Q and R, prediction horizon N_p. The root-mean-square cross track error was plotted against variation of the respective parameters, shown in Fig. 10 and Fig. 11. From the plots, it could be concluded that increasing Q results in better tracking performance, evident from the reduction in the cross-track error. Increasing the weight of the control effort, R, resulted in greater cross-track error and hence, poorer tracking performance.

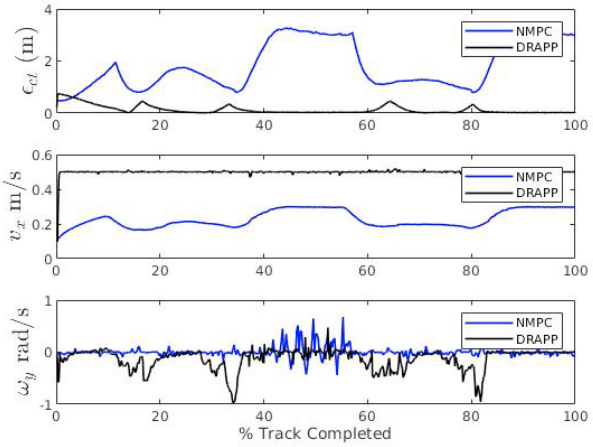

Fig. 4. Vehicle response to rumble strip environment without ω_y consideration for DRAPP

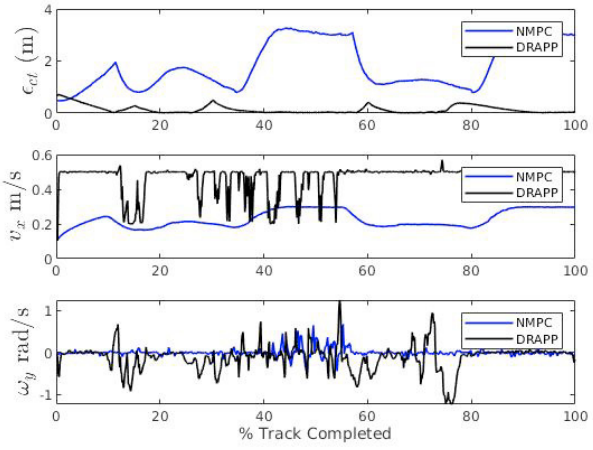

Fig. 5. Vehicle response to rumble strip environment

It was also observed that the cross track error increases when N_p is increased from 0.2 seconds through 2 seconds, keeping the control horizon constant at 0.1 seconds. As expected, the computational time also increases when the prediction horizon increases, due increased number of NMPC optimization iterations.

7. DISCUSSION AND FUTURE WORK

In this study, we explored Deep Reinforcement Adaptive Pure-Pursuit (DRAPP) to enable path(waypoint)-tracking in a skid-steered vehicle operating in an uneven 2.5D terrain. The success of the new control framework was shown for a simulated vehicle under 2 test scenarios consisting of bump disturbances on the terrain. As described in Sec. 6, the DRAPP's waypoint tracking performance was shown to be superior both in terms of higher chosen velocities and lower average cross track error in comparison to the popular Nonlinear Model Predictive Control (NMPC) approach. This performance was achieved without pre-tuning the look ahead distance which was tuned/adapted by DRAPP during runtime. The ability to adapt the preview or look ahead depending on the reference trajectory as well as velocity is a feature that is

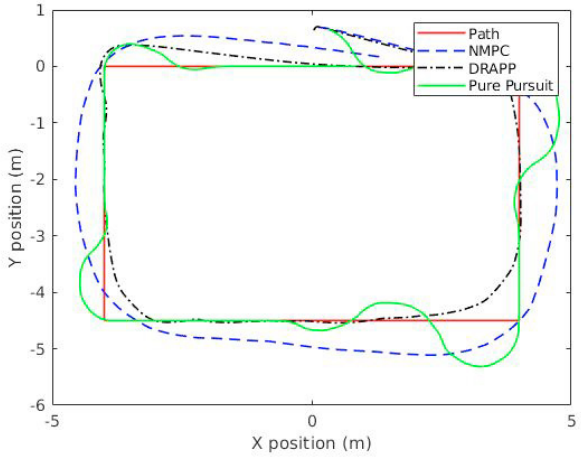

Fig. 6. Vehicle path comparison for rumble strip environment

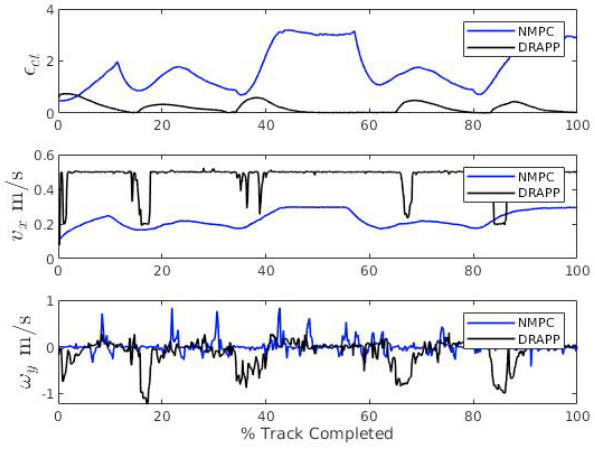

Fig. 7. Vehicle response to uneven bump environment without ω_y consideration for DRAPP

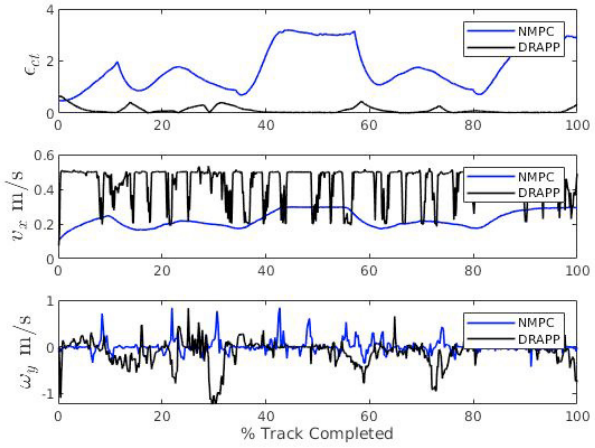

Fig. 8. Vehicle response to uneven bump environment

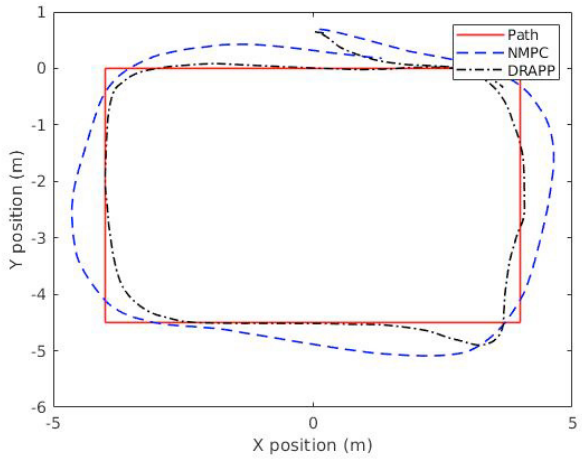

Fig. 9. Vehicle path comparison for uneven bump environment

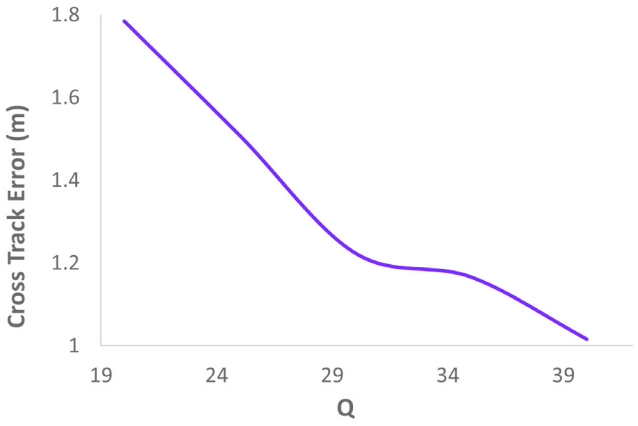

Fig. 10. Cross Track Error (RMS) vs Q

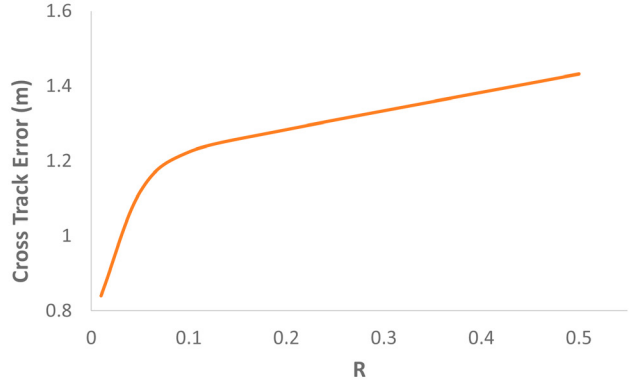

Fig. 11. Cross Track Error (RMS) vs R

absent in NMPC and attempting to add additional constraints to the NMPC problem would significantly increase its complexity. Although the ω_y plots in Figs. 5 and 8 seem like the Z-axis performance of the NMPC is superior, such a conclusion is not credible due to the fact that NMPC is completely blind to the unevenness of the terrain and this result is simply a side-effect of it choosing lower operational speeds. However, the DRAPP is cognizant

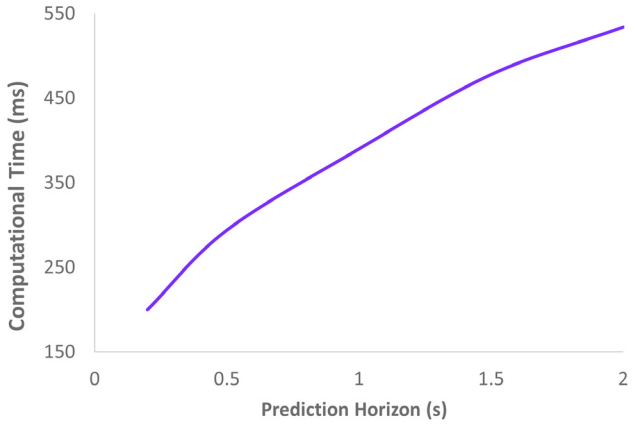

Fig. 12. Computational Time vs Prediction Horizon

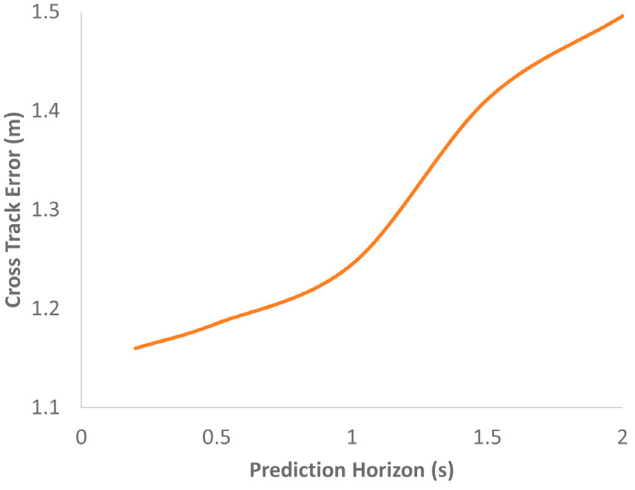

Fig. 13. Cross Track Error (RMS) vs Prediction Horizon

of the unevenness of the terrain and hence results in a very similar ω_y profile despite operating at much higher velocities. Further, this performance can be redesigned to suit our needs by tailoring the objective function (if the Z-axis performance was more important). In addition to the simulated results, a thorough analysis of the objective function used by DRL was presented in order to check for optimality, feasibility and stability. For our future work, we plan to further expand the approach by considering the presence of measurement noise. System-integration with a dynamic lower-level controller is also in the pipeline in order to test dynamic feasibility of the reference generated by this framework. Additionally, an analysis of the real-time performance of the suggested controller using the Clearpath Husky robot platform (Clearpath, 2020) is also in the pipeline. The codebase for the deployment along with the videos of the DRAPP agent performing the maneuvers can be found at Joglekar (2022).

REFERENCES

Aguilera-Marinovic, S., Torres-Torriti, M., and Auat-Cheein, F. (2017). General dynamic model for skid-steer mobile manipulators with wheel–ground interactions. *IEEE/ASME Transactions on Mechatronics*, 22(1), 433–444. doi:10.1109/tmech.2016.2601308.

Andersson, J.A.E., Gillis, J., Horn, G., Rawlings, J.B., and Diehl, M. (2019). CasADi – A software framework for nonlinear optimization and optimal control. *Mathematical Programming Computation*, 11(1), 1–36. doi:10.1007/s12532-018-0139-4.

Chen, I.M. and Chan, C.Y. (2020). Deep reinforcement learning based path tracking controller for autonomous vehicle. *Proceedings of the Institution of Mechanical Engineers, Part D: Journal of Automobile Engineering*, 235, 095440702095459. doi:10.1177/0954407020954591.

Clearpath (2020). Husky - outdoor field research robot by clearpath. URL https://clearpathrobotics.com/husky-unmanned-ground-vehicle-robot/.

Coulter, R.C. (1992). Implementation of the pure pursuit path tracking algorithm. Technical Report CMU-RI-TR-92-01, Carnegie Mellon University, Pittsburgh, PA.

Dankwa, S. and Zheng, W. (2019). Twin-delayed ddpg. *Proceedings of the 3rd International Conference on Vision, Image and Signal Processing*. doi:10.1145/3387168.3387199.

Economou, J., Colyer, R., Tsourdos, A., and White, B. (2002). Fuzzy logic approaches for wheeled skid-steer vehicles. *Proceedings IEEE 56th Vehicular Technology Conference*. doi:10.1109/vetecf.2002.1040749.

Gheisarnejad, M. and Khooban, M.H. (2021). An intelligent non-integer pid controller-based deep reinforcement learning: Implementation and experimental results. *IEEE Transactions on Industrial Electronics*, 68(4), 3609–3618. doi:10.1109/TIE.2020.2979561.

Goel, A. and Chauhan, S. (2021). *Adaptive Look-Ahead Distance for Pure Pursuit Controller with Deep Reinforcement Learning Techniques*. Association for Computing Machinery, New York, NY, USA. URL https://doi.org/10.1145/3478586.3478600.

Joglekar, A. (2022). Code for deep reinforcement learning based adaptation of pure-pursuit path-tracking control for skid-steered vehicles. https://github.com/ajinkya-joglekar/mecc_22_rl_pid.

Joglekar, A., Krovi, V., Brudnak, M., and Smereka, J.M. (2021). Hybrid reinforcement learning based controller for autonomous navigation. *(Accepted, To be published)2021 94th Vehicular Technology Conference (VTC 2021)*.

Khan, R., Mumtaz, F., Raza, A., and Mazhar, N. (2020). Comprehensive study of skid-steer wheeled mobile robots: development and challenges. *Industrial Robot*, ahead-of-print. doi:10.1108/IR-04-2020-0082.

Kim, T., Kim, W., Choi, S., and Jin Kim, H. (2017). Path tracking for a skid-steer vehicle using model predictive control with on-line sparse gaussian process. *IFAC-PapersOnLine*, 50(1), 5755–5760. doi:https://doi.org/10.1016/j.ifacol.2017.08.1140. URL https://www.sciencedirect.com/science/article/pii/S240589631731635X. 20th IFAC World Congress.

Mandow, A., Martinez, J.L., Morales, J., Blanco, J.L., Garcia-Cerezo, A., and Gonzalez, J. (2007). Experimental kinematics for wheeled skid-steer mobile robots. In *2007 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 1222–1227. doi:10.1109/IROS.2007.4399139.

Park, M.W., Lee, S.W., and Han, W.Y. (2014). Development of lateral control system for autonomous vehicle based on adaptive pure pursuit algorithm. *2014 14th International Conference on Control, Automation and Systems (ICCAS 2014)*. doi:10.1109/iccas.2014.6987787.

Rabiee, S. and Biswas, J. (2019). A friction-based kinematic model for skid-steer wheeled mobile robots. In *2019 International Conference on Robotics and Automation (ICRA)*, 8563–8569. doi:10.1109/ICRA.2019.8794216.

Shah, G. and Engell, S. (2011). Tuning mpc for desired closed-loop performance for mimo systems. In *Proceedings of the 2011 American Control Conference*, 4404–4409. doi:10.1109/ACC.2011.5991581.

Slotine, J.J.E. and Li, W. (2005). *Applied nonlinear control*. Pearson Education Taiwan.

Srinivasan, S., Schmid, M.J., and Krovi, V. (2021). Adaptive and reference shaping control for steer-by-wire vehicles in high-speed maneuvers. *IFAC-PapersOnLine*, 54(20), 166–171. doi:10.1016/j.ifacol.2021.11.170.

Sukhil, V. and Behl, M. (2021). Adaptive lookahead pure-pursuit for autonomous racing. *CoRR*, abs/2111.08873. URL https://arxiv.org/abs/2111.08873.

Tang, L., Yan, F., Zou, B., Wang, K., and Lv, C. (2020). An improved kinematic model predictive control for high-speed path tracking of autonomous vehicles. *IEEE Access*, 8, 51400–51413. doi:10.1109/ACCESS.2020.2980188.

Wächter, A. and Biegler, L.T. (2006). On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. *Mathematical Programming*, 106(1), 25–57. doi:10.1007/s10107-004-0559-y. URL https://doi.org/10.1007/s10107-004-0559-y.

Zhao, Z., Liu, H., Chen, H., Hu, J., and Guo, H. (2019). Kinematics-aware model predictive control for autonomous high-speed tracked vehicles under the off-road conditions. *Mechanical Systems and Signal Processing*, 123, 333–350. doi:https://doi.org/10.1016/j.ymssp.2019.01.005. URL https://www.sciencedirect.com/science/article/pii/S0888327019300068.