Enhancing Mobile Robot Trajectory Tracking with Time Series Regression and Reinforcement Learning

Usama Habib
Department of Computer Science
University of Sharjah
P.O.Box. 27272, Sharjah, UAE
U23102906@sharjah.ac.ae

Ghaith Jamjoum
Department of Computer Science
University of Sharjah
P.O.Box. 27272, Sharjah, UAE
U23102953@sharjah.ac.ae

***Abstract*—The application of robotics has enhanced many sectors by delivering robust performance in tasks with minimum accessibility, making it very efficient without much human interference. In the field of robotics, the control of a robot has become an active area of interest and technological adoption. Many Active Disturbance Rejection Controller (ADRC) designs have widely been adopted for their wide applications, ranging from disturbance rejection to state estimation, based on the type of controller applied. The manual tuning process of controller parameters highlights the need for automation because it is time-consuming. This paper presents an innovative approach to tune the parameters of ADRC using Time Series Regression (TSR) and Reinforcement Learning (RL) based on Deep Deterministic Policy Gradient (DDPG) for a Differential Drive Mobile Robot (DDMR). The RL agent learn the ideal parameters of ADRC controller through iterative interactions with a simulated environment where regression provides future estimates of the position to RL agent, improving ADRC's performance without the need of manual tuning. The effectiveness of the suggested concept that enhances trajectory tracking performance is demonstrated by the simulation results. Regression and reinforcement learning are combined to create a promising automated controller tuning solution that could lead to more intelligent and adaptable robotic systems.**

***Index Terms*—Artificial Intelligence (AI), Reinforcement Learning (RL), Time Series Regression (TSR), Active Disturbance Rejection Control (ADRC), Differential Drive Mobile Robot (DDMR), Deep Deterministic Policy Gradient (DDPG)**

I. Introduction

The addition of Artificial Intelligence (AI) into control systems has overcome tradition control techniques and can handle more complex and nonlinear environments. Comparing different AI approaches, the Reinforcement Learning (RL) proved its significant potential in improving the control strategy by interacting to the environment. RL is especially helpful in scenarios where the system dynamics are hard to accurately model or are subject to change, in contrast to classical control techniques that depend on exact mathematical models. RL allows data-driven learning, which is especially helpful in scenarios where the system dynamics are hard to accurately model or are subject to change, in contrast to classical control techniques that depend on exact mathematical models [1].

RL is a type of machine learning (ML) where an agent learns to make decisions by receiving feedback in the form of rewards or penalties [2]. In control systems, RL helps by autonomously tuning controller parameters, learning optimal control policies, and adapting to varying conditions without human intervention. This capability is particularly valuable for robots that need to operate in dynamic, uncertain, or complex environments. Particularly, Deep Reinforcement Learning without taking precise models of the system into account is transforming control systems especially by handling non-linear dynamics. The authors in [3] explained that RL model is based on interactions with the system, feedback and reward optimizing the control policy, which is applicable to the real world problem especially where system contains uncertainty.

Regression is a statistical ML technique that models the relationship between one or more independent variables and dependent variable. Finding patterns in data, regression has wide applications in prediction tasks such as demand forecasting, trend analysis, and anomaly detection. Time series regression is a specialized form of regression in which data points are collected at regular time intervalsv [4]. It considers temporal dependencies and makes it particularly valuable in applications involving stock price prediction, weather forecasting and industrial process monitoring. In control systems, regression can play an important role in the identification of system dynamics, optimisation of control parameters and modeling input-output relationships aiming at enhancement of stability and performance. For example, with regression models, one is able to predict the system responses to changes in inputs-allowing for adaptive and even predictive control strategies.

In industry and robotics, PID controllers are widely used because they are simple and effective in controlling linear processes. Their operation is to provide a control signal proportional to a weighted sum of the control error and its integral and derivative. Although their applications range from temperature control and motor speed to robotic arm movement, these controllers are not robust to external disturbances and model uncertainties. To overcome these shortcomings, Active Disturbance Rejection Control, dynamically copes with both internal and external disturbances. The basic components of ADRC are the Tracking Differentiator (TD) for input signal smoothing, the Extended State Observer (ESO) for estimating states and disturbances, and the Nonlinear State Error Feedback (NLSEF) for stabilization. However, it is

difficult to tune ADRC parameters manually, especially for nonlinear systems. Adaptive control becomes critical when fixed-parameter controllers, such as PID, fail in changing environments. Robots often face such unpredictable scenarios-rough terrain or shifts in payload, for example-where adaptive control ensures stability and accuracy.

The remainder of the paper is organized as follows, section 2 is on literature review, section 3 provides system description and modeling, followed by section 4 that briefs the control design, section 5 presents results and discussion, section 6 concludes the paper.

II. Related Works

The authors in [5] implemented Twin Delayed Deep Deterministic Policy Gradient (TD3) on a PyBullet Racecar model in the OpenAI Gym environment. They used a reward function that stabilizes the policy convergence for differential drive robot. The system is trained in such a way that it maintains a distance from walls and move towards the goal by avoiding collisions. In another work, the ability of RL-based control was to identify the best solution and retrieve the best control policy, as one of its qualities that makes it compatible with conventional control methods. The best course of action is followed by interacting with the environment, getting feedback in the form of rewards or penalties, and learning how to act in a way that maximizes cumulative rewards over time [6]. In [7], an optimized nonlinear tracking control method using an Actor-Critic Reinforcement Learning Strategy is proposed. This approach involves a neural network for both actor and critic to handle non-linear dynamic systems, where convergence is guaranteed using Lyapunov stability theory.

Jingqing Han [8] developed Active Disturbance Rejection Control (ADRC) in the 1990s, and it has since been investigated and applied in a variety of real-world fields, such as robotics. The ADRC is designed to dynamically reject internal and external disturbances to accurately control the system. Unlike classical PID controllers, ADRC adapts disturbances in real-time rather than relying on fixed gains and visualizing a model of the system. ADRC consist of three components: Tracking Differentiator (TD) to smooth the desired input signal, Extended State Observer (ESO) to estimate the system states and handle unknown disturbances and the Nonlinear State Error Feedback (NLSEF) which generate control signals that stabilize the system [9]. PID was also used to control mobile robots [10], they doesn't require a mathematical model of the system and are sensitive to the environmental variations and it cannot handle uncertainties and disturbance like ADRC. but it is based on fixed parameters and struggle with changing dynamics, ADRC overcomes PID by not requiring a perfect mathematical model of the system and handle disturbances better [11].

Authors in [12] proposed a framework for mobile manipulator system and achieves separate trajectory tracking for both. The appropriate ADRC controller is designed, and the nonlinear Extended State Observer (ESO) disturbance is applied to estimate and compensate for disturbance. The ADRC approach's resilience in managing MIMO nonlinear systems with time-varying dynamics shown in [12] by the simulation results. In [13] [14], controller parameters has been scaled so that the tuning parameters can be reduced to only one, by developing ADRC block library in MATLAB/Simulink, enabling easy construction and customization of ADRC systems using graphical modeling.

The PID parameters and as well as ESO bandwidth require manual tuning. A system with complex dynamics and high non-linearity is hard to tune because its time consuming process, manually tuned parameters can never provide the optimal result especially when external disturbances are varying because such disturbances and non-linearity require dynamic adaption [15]. The contribution of this paper is to employ a Deep Deterministic Policy Gradient (DDPG) algorithm in conjunction with RL and regression to design such a system using an ADRC controller. To guarantee accurate estimation and tracking, regression provides future position based on data and the RL agent automatically adjusts the ADRC controller's parameters and trains to converge to the ideal values. This research builds on this that the developed controller is able to provide better trajectory tracking and can be extended and applied in various application.

III. Methodology

A differential drive robot is a widely used mobile robot configuration, especially in indoor applications and research, due to its simplicity and control efficiency. By altering the angular velocities of the two independently driven wheels on either side of the robot, it enables omnidirectional maneuverability. Rotating around its central axis and moving forward and backward are made possible by varying the speeds of the left and right wheels.

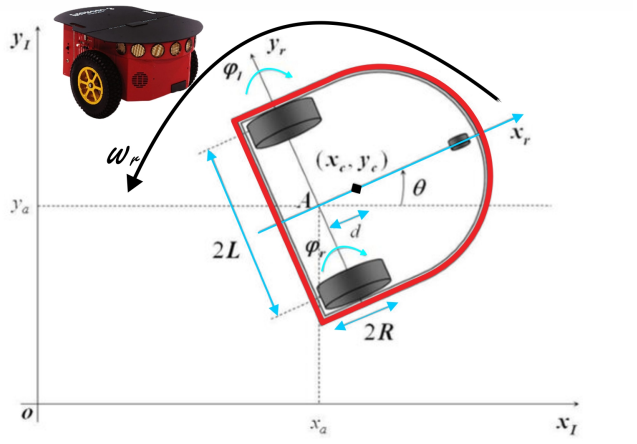

Fig. 1: DDMR Platform

Figure 1 illustrates the geometric properties related to DDMR, (x_c, y_c) are the coordinates of the robot's center point in the reference frame, $2R$ is the distance between the center of the wheels and the point A on the arc path, $2L$ represents the distance between the two wheels, ϕ_r the rotation angle of

the right wheel, ϕ_l is the rotation angle of the left wheel, d the distance between the center point (x_c, y_c) and the point A along the longitudinal axis, (x_r, y_r) are the coordinates of the target/reference point, (x_a, y_a) are the axes representing the global reference frame.

A. Kinematics Model

DDMR's inverse kinematics equation is to convert desired trajectory (x_r, y_r) into angular velocities of left $\dot{\phi}_l$ wheel, right $\dot{\phi}_r$ wheel and θ_r [16]. As shown below:

$$v = \sqrt{{dx_r}^2 + {dy_r}^2} \tag{1}$$

$$w = \frac{d\dot{y}_r \cdot dx_r - dy_r \cdot d\dot{x}_r}{v^2} \tag{2}$$

$$\theta = \arctan\left(\frac{dy_r}{dx_r}\right) \tag{3}$$

$$\theta_{\text{corr}} = \text{atan2}(e_y, e_x) \tag{4}$$

$$\theta = \theta_{\text{des}} + (\theta_{\text{corr}} - \theta_{\text{des}}) \tag{5}$$

The desired angular velocities of the left and right wheels are calculated as:

$$\dot{\phi}_{dl} = \frac{2v - wL}{2R} \tag{6}$$

$$\dot{\phi}_{dr} = \frac{2v + wL}{2R} \tag{7}$$

where L is distance between the wheels, R is radius of each wheel, v linear velocity and w angular velocity of the robot, derived from dx_r and dy_r velocities and acceleration $d\dot{x}_r$ and $d\dot{y}_r$.

First velocities in x and y directions (dx_r, dy_r) are calculated, then linear velocity v and angular velocity w are calculated using Equation (1) and (9), leading to left and right desired angular velocities $(\dot{\phi}_{dl}, \dot{\phi}_{dr})$. The error between desired and final trajectory (e_x, e_y) using equation 5 to be feed directly to the kinematics block. Following are the parameters used in the above equations.

The forward kinematics model of the system is converting actual angular velocities $(\dot{\phi}_{al}, \dot{\phi}_{ar})$ of the wheel into the robot's trajectory (x_a, y_a) [16], then the feed back to inverse kinematics for trajectory error calculation, as shown in the equations below.

$$v = \frac{R(\dot{\phi}_{al} + \dot{\phi}_{ar})}{2} \tag{8}$$

$$w = \frac{R(\dot{\phi}_{al} - \dot{\phi}_{ar})}{L} \tag{9}$$

The (x_a, y_a) positions are then calculated as:

$$x_a = v\cos(\theta_a) \tag{10}$$

$$y_a = v\sin(\theta_a) \tag{11}$$

θ_a in equation (10) and (11) is calculated in dynamics block using equation (9) and then passed to integrator, this provides direction to actual trajectory (x_a, y_a). The kinematic model depends on $(\dot{\phi}_{al}, \dot{\phi}_{ar})$ which is the output of dynamic model explained in the below section. Figure 2 provides an overall architecture of the system.

B. Dynamics Model

The purpose of the dynamic model of the system is to show how the input torque effects the output of the robot. The input to the dynamic model is torque of left and right motors τ_{l} and τ_{r}, and output is the angular velocities of left and right wheels [17]. Also, the actual orientation angle θ_a of the robot is calculated in dynamics block. In a simplified form, the dynamic model equation's derivation is as follows:

$$M(q)\ddot{q} + V(q,\dot{q})\dot{q} = B(q)\tau - \Lambda^T(q)\cdot\lambda \tag{12}$$

Here, $M(q) \in \mathbb{R}^{n\times n}$ is inertia matrix, which includes terms related to the robot's mass m, wheel radius R, distance between the wheels L, wheel inertia I_w, and the robot's moment of inertia I. $V(q,\dot{q}) \in \mathbb{R}^{2\times 2}$ is coriolis matrix, with terms including the mass m, distance d from the center of mass to the wheels, and the robot's angular velocity $\dot{\theta}$. $B(q) \in \mathbb{R}^{2\times 2}$ is input matrix, which maps the torque inputs to the system's motion, involving R wheel radius and L distance between the wheels. $\eta \in \mathbb{R}^{2\times 1}$ vector of angular velocities of the left and right wheels. Where $\tau \in \mathbb{R}^{2\times 1}$ is torque input vector applied to the wheels. Building upon equation (13) from [16].

$$\dot{q} = S(q)\eta \tag{13}$$

Given that the transformation matrix $S(q)$ is denoted as the null space of the constraint matrix $\Lambda(q)$, then the following holds:

$$S^T(q)\Lambda^T(q) = 0 \tag{14}$$

By obtaining the derivative of Equation (11) with respect to time, we get:

$$\ddot{q} = \dot{S}(q)\eta + S(q)\dot{\eta} \tag{15}$$

Recalling Equation (10) and substituting both (11) and (13) into it, we obtain dynamic equation in reduced form:

$$M(q)\dot{\eta} + V(q,\dot{q})\eta = B(q)\tau \tag{16}$$

The parameters in the equation are listed below.

$$M(q) = \begin{bmatrix} \frac{R^2 I_w + L^2 m}{4} + I & -\frac{L^2 m}{4} + I \\ -\frac{L^2 m}{4} + I & \frac{R^2 I_w + L^2 m}{4} + I \end{bmatrix},$$

$$V(q,\dot{q}) = \begin{bmatrix} 0 & -\frac{md\dot{\theta}}{2} \\ \frac{md\dot{\theta}}{2} & 0 \end{bmatrix},$$

$$B(q) = \begin{bmatrix} \frac{1}{R} & \frac{1}{R} \\ \frac{L}{2R} & -\frac{L}{2R} \end{bmatrix}, \eta = \begin{bmatrix} \dot{\varphi}_r \\ \dot{\varphi}_l \end{bmatrix}, \tau = \begin{bmatrix} \tau_r \\ \tau_l \end{bmatrix}.$$

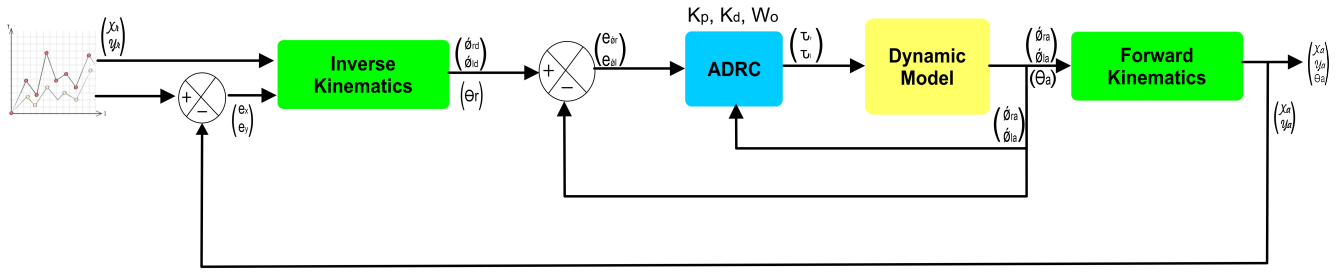

Fig. 2: Architecture Overview

C. Controller Design

Figure 2 contains the block diagram of the system. Where the system begins with a desired trajectory (x_r, y_r), which is feed to the inverse kinematics block, the block results in angular left and right velocities which will be controlled throughout the system $(\dot{\phi}_{dl}, \dot{\phi}_{dr})$. After subtracting angular velocities desired from the actual, the ADRC block receives the error signals as in equation (17) and (18).

$$e_{\dot{\phi}_r} = \dot{\phi}_{d_r} - \dot{\phi}_{a_r} \tag{17}$$

$$e_{\dot{\phi}_l} = \dot{\phi}_{d_l} - \dot{\phi}_{a_l} \tag{18}$$

By receiving the torque values from controller (τ_r, τ_l), dynamic model is simulating the robot itself, in which we can feed it with torque, and measure the both left and right actual angular velocities $(\dot{\phi}_{al}, \dot{\phi}_{ar})$ of the wheels, which are then converted into robots (x_a, y_a) trajectory using forward kinematics block.

D. Time series Regression

In this project, Time Series Regresion, is used to predict future x and y positions along with orientation angle $theta$. Long Short Term Memory (LSTM) model is used to help RL agent to know (t+1) the future state of the robot based on simulation. To train LSTM model data is collected through a simulation model based on multiple iterations involving simple control and applying different disturbances such as step, sinusoidal and pulse signals which replicate real disturbances. The data contained 7 features: left angular velocity, right angular velocity, x-axis , y-axis postions, orientation angle θ also left and right control torque signals. These iterations end up with 12404 rows for each feature. Additionally, the inclusion of disturbances in the training data helps the LSTM model generalize well to scenarios with varying external influences. A sample of the first five rows of the dataset is in figure 3.

The dataset is preprocessed by normalizing all input and output features using Min-Max Scaling to ensure numerical stability during training. Input features are organized into sequences of length 10, with the corresponding outputs representing the robot's future x, y, and θ values at . The data is split into three sets: a training set (70%) used for model learning, a validation set (15%) used for hyperparameter tuning and performance monitoring, and a testing set (15%) used for evaluating the model's performance on unseen data.

WL_Real	WR_Real	Theta_Real	X_Real	Y_Real	TorqueL	TorqueR
0.794701987	1.296619031	0	0	0	0.07419257	0.100485018
0.793887084	1.295289396	0	0.000993037	0	0.074095913	0.100394587
0.7929198	1.29371109	0.001986755	0.001984826	9.85042E-07	0.073985097	0.100284843
0.792065602	1.292317257	0.003971472	0.002975418	3.93596E-06	0.073884916	0.100189348
0.791403299	1.29123649	0.005953771	0.003965038	8.84697E-06	0.073802709	0.10011808

Fig. 3: Regression Data

The LSTM architecture consists of an input layer that accepts sequences of size equal to the number of input features, an LSTM layer with 50 hidden units, a fully connected layer that maps hidden states to the three output features (x, y, θ), and a regression layer that computes the loss between predicted and actual values during training. The LSTM model is trained using the Adam optimizer with a learning rate of 0.001. A mini-batch size of 32 is used for efficient computation, and the training process is monitored using the validation dataset. Training progress is visualized with plots showing loss over epochs to ensure convergence, as shown in figure 4.

Further, based on the testing of LSTM model, RMSE for first feature x-axis position is 0.0035077, for y-axis position is 0.0018348 and for orientation angle θ is 0.015658. Which indicates that testing error is less than the training error, so the model has good generalization low variance, further the predictedand values based on testing set is predicted almost same to the testing set, hence, the model has low bias as well.

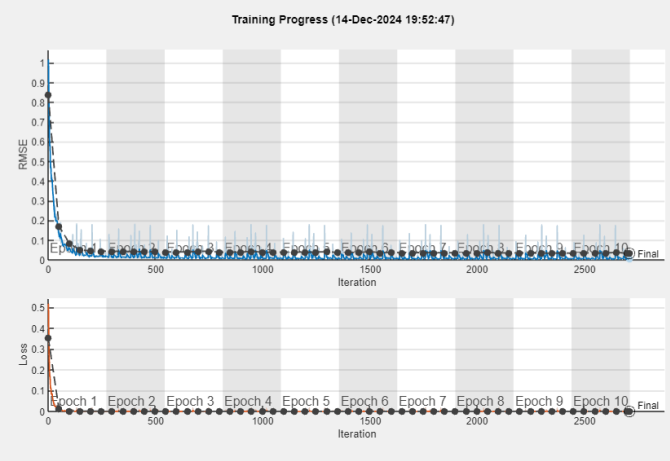

Fig. 4: Regression Plot

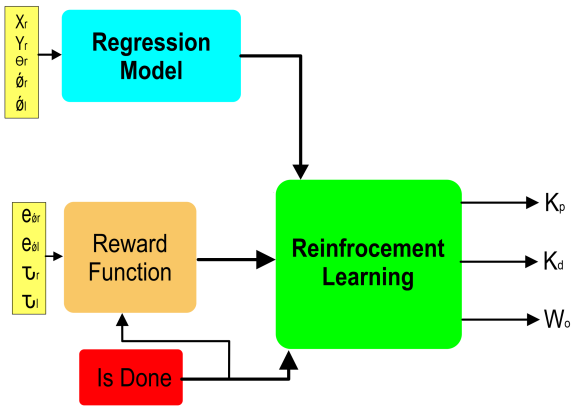

Fig. 5: RL and Regression Structure

E. Reinforcement Learning

RL has shown itself to be a useful tool for automating controllers in recent years [1]. Without the use of mathematical models, RL allows controllers to learn the best control strategies through trial-and-error interactions with the system. The DDPG algorithm works well for continuous action spaces, such as PID gains. The critic network combines the state and action inputs, processes them in parallel, and then combines them in a common path.

Observation for the RL agent consist of desired right and left angular velocities ($\dot{\phi}_{dl}$, $\dot{\phi}_{dr}$), orientation angle (θ_r), errors ($e_{\dot{\phi}_l}$, $e_{\dot{\phi}_r}$) and derivative of the errors in angular velocities ($\dot{e}_{\dot{\phi}_l}$, $\dot{e}_{\dot{\phi}_r}$). Further, the reward function plays a crucial role in RL agent learning process. This function guides the agent during learning that the step taken is either positive or negative which is called rewards, a positive reward encourage the behavior that are favorable to the objective/goal, while negative are opposite. This function is a combination of parameters of the system which are related to actuators. Following is the reward function developed in this paper for DDMR.

$$R = -\alpha \cdot (e_{\dot{\varphi}_r} + e_{\dot{\varphi}_r}) - \beta \cdot \left(\tau_R^2 + \tau_L^2\right) - \gamma \cdot \text{Boundry} \quad (19)$$

Here, $\alpha = 0.001$, $\beta = 0.002$ and $\gamma = 1$, errors($e_{\dot{\phi}_l} + e_{\dot{\phi}_r}$) are angular velocities error penalty term for torque is added based on τ_L^2 and τ_R^2 and boundary term to encourage the agent to stay within desired operating conditions.The general architecture of RL block is shown in figure 5.

IV. Results and Discussion

In this section, the simulation results of regression and RL tuned ADRC are discussed. The experiment is simulated on MATLAB/Simulink. A 30-second circular trajectory is provided to test the controller. The actor network uses a fully connected structure with two layers of 400 neurons each for the state input, culminating in the action layer with three outputs. While critic network contains two layers of 500 neurons. The selected hyper parameters for the RL agent training are shown in Table 1. Beside the learning algorithm hyper parameters also play a significant impact on the training of RL agent. The discount factor is set as 0.99 to have a stable learning, the noise has a lower decay at $1e-4$. So, that exploration should be for large period of training. Furthermore, total number of observations are 7 and actions are 3.

Discount Factor	0.99
Noise Standard Deviation	0.3
Noise Decay Rate	$1e^{-}4$
Actor Learning Rate	$1e^{-}4$
Critic Learning Rate	$3e^{-}4$
Experience Replay Buffer Size	$1e^{6}$
Mini-batch Size	128

TABLE I: Training Hyper-parameters

Figure 5 shows the x, y & θ tracking of desired input trajectory, robot's actual trajectory using ADRC and the proposed ADRC technique for a circular motion. The controller operates with parameter values set as Kp= 150, Kd= 150 and wo= 50. By designing an effective reward function, the objective of the controller is achieved and proven through simulation results. It can be observed from the figures that estimation of the trajectory is almost perfect, where trajectories overlap entirely. This signifies that ESO was tell tuned. Also, proposed ADRC technique is able to follow desired trajectory . Figure 5 also contain trajectory error for x axis, y axis and θ for ADRC and RL tuned ADRC. The trajectory errors are almost negligible, which means that both controllers track the desired trajectory without much deviation in both X and Y axes.

Figure 6 illustrates the angular velocities of the left and right wheels for the desired trajectory, the ADRC controller, and the propsoed ADRC controller. As shown, the signals exhibit initial oscillations before stabilizing and then closely following the desired angular velocities. The results indicate both technique follow the desired angular velocity but proposed ADRC has negligible oscillation as compared to manual tuned ADRC.

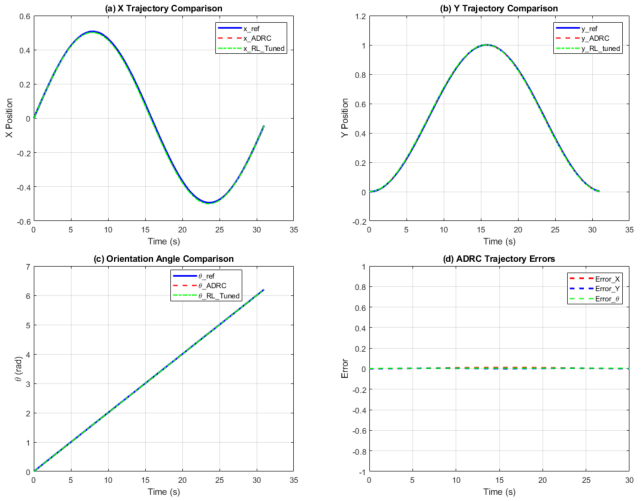

Fig. 6: Trajectory Tracking. (a) X-axis (b) Y-axis (c) θ (d) X, Y & θ trajectory errors

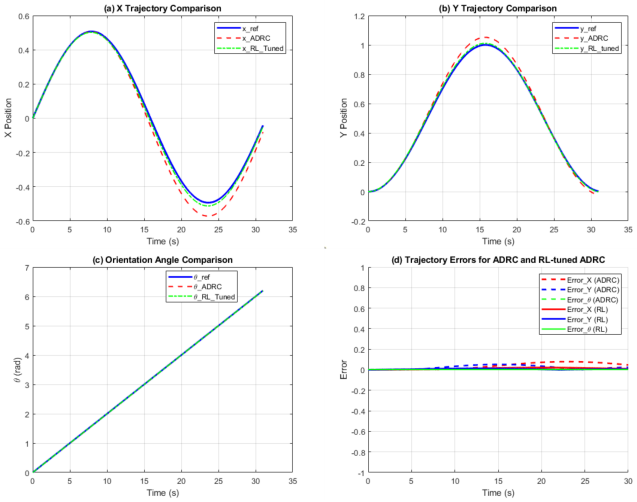

Fig. 8: Trajectory Tracking. (a) X-axis (b) Y-axis (c) θ (d) X, Y & θ trajectory errors after disturbance

Figure 8 contains left and right angular velocity errors after adding step disturbance, regression and RL tuned ADRC slightly increasing angular velocities as compared to manual tuned ADRC to keep the robot on circular path.

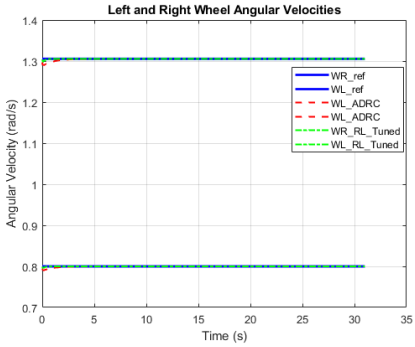

Fig. 7: Left & Right Wheel Angular Velocities

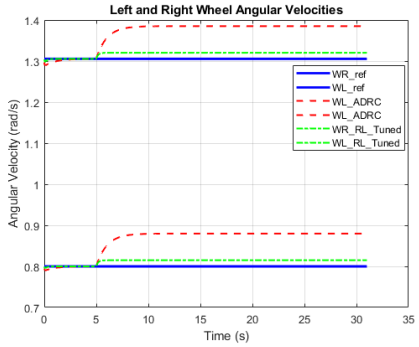

Fig. 9: Left & Right Wheel Angular Velocities after Disturbance

Figure 7 shows the x, y & θ trajectory tracking and errors after adding a step disturbance at t=5 seconds with the magnitude of 1. Its clearly visible in the graph as the continuous disturbance appears the robot start to deviate from the trajectory, ADRC is able to control the disturbance but propsoed approach show better performances by optimizing the parameters. Also the system was tested on pulse disturbance of magnitude 0.5 at t=5 seconds and sinusoidal disturbance of magnitude 0.5 at t=5 seconds, the RL tuned controller performed well as compared to manual tuned ADRC. The trajectory errors of RL tuned ADRC are less then than the manual tuned ADRC, which indicates better performance.

V. Conclusion

The goal of this research is to train a regression model to estimate robot's future position and add this as observations to DDPG-based RL agent to automatically adjust the parameters of an ADRC controller. In order to achieve our main goals, we investigate this hybrid control solution for a differential drive robot, which combines RL approaches with conventional control tactics. Although the ADRC controller is well recognized in industry for its disturbance-rejection capabilities, its parameters are difficult to tune. We can further improve the tracking performance by utilizing an regression and RL agent to learn the best tuning parameters in a dynamic manner. From our simulation results, considering that the system is performing with very accurate trajectory tracking and almost faultless state estimation, we show that ADRC and RL can be combined in an effective way for the purpose of parameter tuning.

In the future, the simulation clearly indicates to lead the research to experiment level, also to build a controller based on only neural network which overcomes the traditional ADRC, with the help of desired trajectory the RL controller result in output torque to control the system.

REFERENCES

[1] J. Shin, T. A. Badgwell, K.-H. Liu, and J. H. Lee, "Reinforcement learning - overview of recent progress and implications for process control," *Computers and Chemical Engineering*, vol. 127, pp. 282–294, 2019.

[2] B. Belousov, H. Abdulsamad, P. Klink, S. Parisi, and J. Peters, Eds., *Reinforcement Learning Algorithms: Analysis and Applications*, ser. Studies in Computational Intelligence. Cham, Switzerland: Springer Nature Switzerland AG, 2021, vol. 883. [Online]. Available: https://doi.org/10.1007/978-3-030-41188-6

[3] D. T. J. K. I. P. Lucian Buşoniu, Tim de Bruin, "Reinforcement learning for control: Performance, stability, and deep approximators," *Annual Reviews in Control*, vol. 46, pp. 8–28, 2018.

[4] R. H. Shumway and D. S. Stoffer, *Time Series Analysis and Its Applications: With R Examples*. New York: Springer Science & Business Media, 2000. [Online]. Available: https://books.google.ae/books?hl=en&lr=&id=8FppWLEFHU8C&oi=fnd&pg=PA8&dq=time+series+regression&ots=b5-uk6WHtr&sig=ugmhi_748PwqkyKLjvy9Dvpa_GA&redir_esc=y#v=onepage&q=time%20series%20regression&f=false

[5] K. F. I. S. A. . Y. A. Mahrukh Shahid, Semab Naimat Khan, "Dynamic goal tracking for differential drive robot using deep reinforcement learning," *Neural Processing Letters*, vol. 55, p. 11559–11576, 2023.

[6] P. Abbeel, "Apprenticeship learning and reinforcement learning with application to robotic control," *Stanford University*, 2008.

[7] G. Wen, C. L. P. Chen, S. S. Ge, H. Yang, and X. Liu, "Optimized adaptive nonlinear tracking control using actor critic reinforcement learning strategy," *IEEE Transactions on Industrial Informatics*, vol. 15, no. 9, pp. 4969–4979, 2019.

[8] G. Feng, L. Huang, and D. Zhu, "A nonlinear auto-disturbance rejection controller for induction motor," in *Proceedings of the 24th Annual Conference of the IEEE Industrial Electronics Society (IECON '98)*, vol. 3. IEEE, 1998, pp. 1509–1514.

[9] G. Herbst, "A simulative study on active disturbance rejection control (adrc) as a control tool for practitioners," *Electronics*, vol. 2, no. 3, pp. 246–279, 2013.

[10] N. H. Thai, T. T. K. Ly, H. Thien, and L. Q. Dzung, "Trajectory tracking control for differential-drive mobile robot by a variable parameter pid controller," *International Journal of Mechanical Engineering and Robotics Research*, vol. 11, no. 8, pp. 614–621, 2022.

[11] M. Drakulic and M. Stankovic, "Adrc-based trajectory tracking of unmanned tracked vehicles under high slippage disturbance," in *2024 14th International Conference on Electrical Engineering (ICEENG)*. IEEE, 2024, pp. 1–6.

[12] Ningyue, Liyan, and Liukeping, "The research of mobile manipulator trajectory tracking cooperative control based on the adrc," in *2015 International Conference on Electrical and Electronic Engineering*. IEEE, 2015, pp. 385–389.

[13] M. Kia, P. Mansouri, and A. Javdani, "Modeling and simulation of a single gain tuning adrc controller in matlab/simulink," *Datis Elevator*, 2021, available online.

[14] Y. Wen-bin and Y. Dong, "Modeling and simulation of an active disturbance rejection controller based on matlab/simulink," *International Journal of Research in Engineering and Science (IJRES)*, vol. 3, no. 7, pp. 62–69, 2015. [Online]. Available: http://www.ijres.org/

[15] C. Fu and W. Tan, "Tuning of linear adrc with known plant information," *ISA Transactions*, vol. 65, pp. 384–393, 2016.

[16] M. Nasir, "Dynamic modeling and control of wheeled mobile robot using active disturbance rejection control for trajectory tracking," University of Sharjah, Tech. Rep., 2024, accessed on 26 October 2024.

[17] R. Dhaouadi and A. A. Hatab, "Dynamic modelling of differential-drive mobile robots using lagrange and newton-euler methodologies: A unified framework," *Advances in Robotics & Automation*, vol. 2, p. 107, 2013, accessed on 26 October 2024. [Online]. Available: http://dx.doi.org/10.4172/2168-9695.1000107