Epersist: A Self Balancing Robot Using PID Controller And Deep Reinforcement Learning

Ghanta Sai Krishna¹ *, Garika Akshay¹, Dyavat Sumith¹
¹Department of Data Science and Artificial Intelligence, IIIT Naya Raipur, Chattisgarh, 492101, India
²Student Member, IEEE

Abstract—A two-wheeled self-balancing robot is an example of an inverse pendulum and is an inherently non-linear, unstable system. The fundamental concept of the proposed framework “Epersist” is to overcome the challenge of counterbalancing an initially unstable system by delivering robust control mechanisms, Proportional Integral Derivative (PID), and Reinforcement Learning (RL). Moreover, the micro-controller NodeMCU ESP32 and inertial sensor in the Epersist employ fewer computational procedures to give accurate instruction regarding the spin of wheels to the motor driver, which helps control the wheels and balance the robot. This framework also consists of the mathematical model of the PID controller and a novel self-trained advantage actor-critic algorithm as the RL agent. After several experiments, control variable calibrations are made as the benchmark values to attain the angle of static equilibrium. This “Epersist” framework proposes PID and RL-assisted functional prototypes and simulations for better utility.

Index Terms—Two Wheeled Self-Balancing Robot, PID, Reinforcement Learning, NodeMCU ESP32

I. INTRODUCTION

The two-wheeled self-balancing robot (TWSBR) is a standard robot with applications in various fields, including transportation and exploration. Over the last few decades, academics and industry have been paying close attention to both the design and regulation of the TWSBR. The TWSBR is considered as a high-order, multi-variable, nonlinear, tightly coupled, inherently unstable system. Conventional methods like PID[1], fuzzy[2], and sliding mode control[3] are proposed in the recent times. Two wheeled mobile robot balancing controller has been tackled as either linearized or as a nonlinear model[4]. The physical characteristics of robot are crucial to achieve optimal control[5]. In practical applications It is frequently desirable to obtain optimality beyond simple stabilisation. Though the existing works had achieved maximum stabilization, they are less capable of achieving optimality. The control systems are not optimal, if the stabilization criterion’s depend physical characteristics of robot. So there might be a need of advanced techniques for achieving optimality.

In recent works, Reinforcement learning (RL) has been included in TWSBR as a control mechanism to attain optimality and achieve stability[6]. The RL enables robots to learn, adapt, and optimize their behaviours by interacting with their surroundings. RL solves the optimization issues that involve an agent interacting with its environment and changing its behaviours or control policies in response to inputs or rewards. To attain better stability, RL techniques are widely used in TWSBR, like Q-learning[7], Proximal Policy Optimization (PPO)[8], and Soft Actor-Critic (SAC)[9]. However, the RL frameworks of existing solutions are based on Q-Learning, PPO and SAC. The proposed methodology highlights the benefits of the advantage Actor-Critic Algorithm (A2C) [10] based self-trained model for the self-balancing robot.

Irrespective of the robot’s control mechanisms, it is essential to understand the theoretical aspects like the transfer function[11], stability[12] etc., which are crucial in judging the system’s stability at a time instance. To understand these theoretical aspects of the robot, simulations of the robot are also required. However, the circumstances and the constraints in executing the hardware prototyping are slightly different from the simulations. Moreover, the hardware experimental data is valuable, and RL architectures usually require millions of data volumes [13]. Apart from the robot’s functionality, it is essential to make it more cost-effective. However, the existing solutions utilize micro-controllers such as Raspberry-Pi[14], Arduino-UNO [15] etc. The overall cost of the robot by utilizing these micro-controllers is expensive. Thus there is a need to improve the cost-efficiency and optimal utility of the system. To overcome all these disadvantages in the existing solutions, the proposed methodology effectively analyses both theoretical aspects via simulations and the hardware prototyping conditions of the robot.

Moreover, the proposed solution utilizes Node-MCU ESP-8266 as a micro-controller to make a cost-effective system. To improve the practical utility of the robot, the proposed solution also consists of a mobile interface which connects to the micro-controller. The significant contributions of the paper are as follows:

The significant contributions and improvements from existing works of Epersist are as follows:

- Analyzing and deriving the theoretical aspects and effects of the PID control mechanism on the robot.
- Building a novel, robust and less computational A2C algorithm-based Deep Reinforcement Learning agent.
- Deploying the PID mechanism and RL model in the NodeMCU to build a cost-effective hardware prototype.
- Deploying a Bluetooth-based mobile application to control the robot for greater utility

II. METHODOLOGY OF EPERIST

This section discusses the procedures involved in the proposed methodology, which is categorized into 2 phases- Physical modelling and PID control mechanism; Deep RL agent. The overview of the proposed methodology is represented in Fig. [1]
calculated the 3-axis gyroscopic \((g_x, g_y, g_z)\) and 3-axis accelerometer \((a_x, a_y, a_z)\) measures. Computationally, the pitch and yaw calculation is based on the sensor measures, which is different from the conventional mathematical model. The instantaneous pitch angle \(\phi\) is dependent on the previous pitch angle \(\dot{\phi}(t = 0)\) and is calculated with a complementary filter as derived in eq. \(5\)

\[
\phi = [\alpha(\dot{\phi} + g_x)] + [(1 - \alpha)(\text{ATAN2}(a_y, a_z))] \tag{5}
\]

Where \(\alpha\) is the filter coefficient, and \(\text{ATAN2}(a_y, a_z)\) represents the angle of \((a_y, a_z)\) in the plane. The error \(\epsilon(t)\) at an instance \(t\) is the difference between the current pitch angle and target pitch angle. This feedback error is applied to the PID control mechanism at every time instance as derived in eq. \(6\). Finally, this is responsible to generate the command for instructing motor driver. For balancing the robot rigidly, the physical parameters play a crucial position. The PID calibration (tuning the \(K_p, K_i, K_d\) values) is performed with trial-error method.

\[
\text{Output} = K_p e(t) + K_i \int e(t) dt + K_d \frac{de(t)}{dt} \tag{6}
\]

**B. The Deep Reinforcement Learning Agent**

The novel self-trained deep reinforcement learning agent is based on the advantage Actor-Critic (A2C) algorithm, which consists of two dependent and similar neural networks (actor and critic) and integrates RL’s policy and value algorithms. The actor (policy function) decides an action at each iteration, and the critic (value function) estimates the quality or the value index of a delivered initial state.
state, the actor decides the further action. Similarly, based on the previous action (rewards) and the state, the critic will produce the quality value of that action. The actor will learn from the temporal difference error (TD), calculated by the advantage function in each nested iteration. The detailed mathematical functionality of the A2C algorithm is characterised in algorithm II-B where the initial yaw and pitch values are derived in the form of weights $Y$ and $\theta$.

**Algorithm 1** A2C Algorithm

Input : Initialize actor ANN $(V^a(s))$, critic ANN $(\pi^\theta(s))$
Initialize environment $E$
for Episode = 1, $M$ do
    Acquire initial observation state $s_0$ from $E$
    for $t = 0, T$ do
        current policy $= a_t \sim \pi(a|\mu, \sigma) = N(\mu|\mu, \sigma)$
        From $E$: execute action $a_t$ and observe reward $= y$ and next state $\rightarrow s_{t+1}$
        Set TD target $= y = y + y.V^a(s_{t+1})$
        By minimizing the loss update critic $= \delta_t = (y_t - V^a(s_t))^2$
        By minimizing the loss update the actor policy: $LOSS = -\log(N(\mu(s_t), \sigma(s_t)))$
        Update $s_t \leftarrow s$
    end
end

**III. EXPERIMENTAL RESULTS**

In this section, the experimental results are presented and analysed. Firstly, the physical modelling and PID tuning are performed based on our system design in the MATLAB Simulink. The RL model for the robot is designed in the Pybullet-OpenAI environment and further injected to the micro-controller. The physical parameters of the robot are shown in Table 1.

| Physical Parameter       | Value/Unit |
|-------------------------|------------|
| Mass of main body       | 135 g      |
| Mass of pendulum        | 60 g       |
| Diameter of Wheel       | 5 cm       |
| Distance between wheels | 20 cm      |
| Static Friction b/w Surfaces | 1.15     |

Table 1. Physical parameters of Epersist Robot

The hardware prototype proposed is cost-effective and consists of NodeMCU ESP32, IMU MPU 6050, H-1298N motor bridge, 6V gear motors, and a self-designed 3D printed base plate. The NodeMCU ESP32 acts as the micro-controller, in which the RL model or PID mechanism is manually uploaded. The injection of the RL model (.h5 extension) to NodeMCU ESP32 is one of the challenging tasks, which is handled precisely. This micro-controller is compatible with connecting with a mobile phone via Bluetooth. An interactive mobile application is built to provide the robot with initial control variables (pitch, yaw) as shown in Fig. . These interactions between the mobile application and the robot are efficient. An Inertial Measurement Unit (6-axis IMU MPU 6050) sensor is utilised to acquire the 3-axis accelerometer and 3-axis gyroscopic measures. These six measures are responsible for the calculation of the pitch (tilt angle) and yaw (horizontal movement) of the robot.

**Fig. 5. Epersist Robot and the Mobile Interface**

The optimal calibration points must be identified to calculate the target values of control variables and sensor measures. After several experimentation’s, the $K_p$, $K_i$ and $K_d$ values are tuned and are calibrated to 1970, 21950, 19.5 respectively. Furthermore, the calibrated offsets of the sensor measures are shown in Table 2.

| Offset               | Value   |
|----------------------|---------|
| X Accelerometer Offset | -1780   |
| Y Accelerometer Offset | 750     |
| Z Accelerometer Offset | 2700    |
| X Gyroscopic Offset  | 180     |
| Y Gyroscopic Offset  | 76      |
| Z Gyroscopic Offset  | 61      |

Table 2. Offset measures for the IMU MPU 6050 Sensor

Irrespective of the system’s physical characteristics, the trained RL model can be utilized for our robot. As neural networks are involved in the RL model, the training and uploading time to NodeMCU is longer than the PID mechanism. The description and summary of the trained RL model are shown in Table 3.

| Parameter | Value |
|-----------|-------|
| Trained Episodes | 7775  |
| Trained Steps   | 1.5e+06 |
| Total Training Time | 19 Hrs |
| Time To Upload | 23.5 Secs |
| Maximum Possible Reward After Training | 60     |
| Maximum Achieved Reward After Training | 56.42  |

Table 3. Summary and Description of RL self-trained RL model

After tuning the $K_p$, $K_i$, $K_d$ and hyper-parameters of both control mechanisms. The performance of the control mechanisms is analyzed with the sensor measures from the functional prototype. The comparative assessment of both control mechanisms with the hardware prototype is demonstrated in Fig. 6. The comparison between the PID and RL is based on the angle of deviation (pitch) and horizontal distance (yaw) covered by the robot over a while.
The final angle of deviation must be approximately zero to call it a balanced system. Our experimentation’s observed that the robot’s overall movement with PID was not as smooth as with RL. The PID-assisted robot’s angle of deviation (pitch) oscillates more than the RL. As the number of oscillations increases, the smoothness of the robot decreases in a unit of time. The mean settling time of the PID robot is 7 seconds, whereas the RL robot takes 5 seconds to achieve maximum stability. Furthermore, the RL robot had covered less distance when compared to the PID robot to achieve maximum stability. The adequate difference between the distance covered by the robot for both control mechanisms is approximately 20 cm. To conclude, the RL control mechanism for the robot is better and more effective than the PID control mechanism in terms of mean settling time and distance travelled in mean settling time.

IV. CONCLUSION

The proposed TWSBR “Epersist” is a flawless end-to-end framework for studies of advanced control techniques due to its complicated task of balancing the structure. The proposed framework has many advantages in terms of the time complexity of the contemporary self-trained RL agent and the cost efficiency of the robot over other existing frameworks. The physical modelling of the system is appropriate, and the outcomes match the experimental results. Initially, the framework is analyzed via simulation (MATLAB for PID, OpenAI for RL), and hardware prototyping is performed with NodeMCU ESP32 microcontroller, IMU MPU 6050 sensor. The PID and RL mechanisms are analyzed from the initial stages based on the physical system design. We represented the performance and experimental results for both control mechanisms. The utility of this robot is simple with the interactive mobile interface, which is connected over Bluetooth. This robot is limited to a random path. The robot cannot move on the desired path. An extension of the desired path can be included in the framework for future work, and the advanced RL control agents can also be included.

REFERENCES

[1] N. T and P. K. T, "PID Controller Based Two Wheeled Self Balancing Robot," 2021 5th International Conference on Trends in Electronics and Informatics (ICOEI), 2021, pp. 1-4, doi: 10.1109/ICOEI51242.2021.9453091.
[2] E. Susanto, A. Surya Wibowo and E. Ghiffary Rachman, "Fuzzy Swing Up Control and Optimal State Feedback Stabilization for Self-Erecting Inverted Pendulum," in IEEE Access, vol. 8, pp. 6496-6504, 2020, doi: 10.1109/ACCESS.2019.2963399.
[3] J. Huang, M. Zhang, S. Ri, C. Xiong, Z. Li and Y. Kang, "High-Order Disturbance-Observer-Based Sliding Mode Control for Mobile Wheeled Inverted Pendulum Systems," in IEEE Transactions on Industrial Electronics, vol. 67, no. 3, pp. 2030-2041, March 2020, doi: 10.1109/TIE.2019.2903778.
[4] S. Kim and S. Kwon, "Nonlinear Optimal Control Design for Underactuated Two-Wheeled Inverted Pendulum Mobile Platform," in IEEE/ASME Transactions on Mechatronics, vol. 22, no. 6, pp. 2803-2808, Dec. 2017, doi: 10.1109/TMECH.2017.2767085.
[5] M. S. Mahmoud and M. T. Nasir, "Robust control design of wheeled inverted pendulum assistant robot," in IEEE/CAA Journal of Automatica Sinica, vol. 4, no. 4, pp. 628-638, 2017, doi: 10.1109/JAS.2017.7510613.
[6] C. Chang and S. Chang, "Using Reinforcement Learning to Achieve Two Wheeled Self Balancing Control," 2016 International Computer Symposium (ICS), 2016, pp. 104-107, doi: 10.1109/ICS.2016.0029.
[7] L. Guo, S. A. A. Rizvi and Z. Lin, "Optimal Control of a Two-Wheeled Self-Balancing Robot by Reinforcement Q-learning," 2020 IEEE 16th International Conference on Control & Automation (ICCA), 2020, pp. 955-960, doi: 10.1109/ICSI1439.2020.9264485.
[8] G. Paczolay and I. Harmati, "A New Advantage Actor-Critic Algorithm For Multi-Agent Environments," 2020 23rd International Symposium on Measurement and Control in Robotics (ISMCR), 2020, pp. 1-6, doi: 10.1109/ISMCR51255.2020.9263738.
[9] Y. Gu, Y. Cheng, C. L. P. Chen and X. Wang, "Proximal Policy Optimization With Policy Feedback," in IEEE Transactions on Systems, Man, and Cybernetics: Systems, vol. 52, no. 7, pp. 4600-4610, July 2022, doi: 10.1109/TSMC.2021.3098451.
[10] Sharma, Sahil. "SAC-RL: Continuous Control of Wheeled Mobile Robot for Navigation in a Dynamic Environment." (2020).
[11] Jian-Shiun Lew and K. B. Lim, "Robust control of identified reduced-interval transfer function," in IEEE Transactions on Control Systems Technology, vol. 8, no. 5, pp. 833-841, Sept. 2000, doi: 10.1109/87.865855.
[12] C. Song, W. Ji, X. Gong and Z. Hu, "Research on stability for linear control system with time delay," 2011 IEEE International Conference on Mechatronics and Automation, 2011, pp. 2428-2432, doi: 10.1109/ICMA.2011.5986332.
[13] E. Li et al., "Model Learning for Two-Wheeled Robot Self-Balance Control," 2019 IEEE International Conference on Robotics and Biomimetics (ROBIO), 2019, pp. 1582-1587, doi: 10.1109/ROBIO49542.2019.8961382.
[14] F. F. Rabbany, A. Qurthobi and A. Suhendi, "Design of Self-Balancing Virtual Reality Robot Using PID Control Method and Complementary Filter," 2021 IEEE International Conference on Robotics and Biomimetics (ROBIO), 2019, pp. 1582-1587, doi: 10.1109/ROBIO49542.2019.8961382.
[15] A. S. Shekhawat and Y. Rohilla, "Design and Control of Two-wheeled Self-Balancing Robot Using Arduino," 2020 International Conference on Smart Electronics and Communication (ICOSEC), 2020, pp. 1025-1030, doi: 10.1109/ICOSEC49089.2020.9215421.