Design and Navigation of a Self-Balancing Two-Wheeled Robot in Uneven Terrain

Ayush Chaudhary

120429125

Abstract

This report details the development of a self-balancing two-wheeled robot, *BalanceBot*, using PyBullet and Gymnasium. The robot uses a PID controller for stabilization and Proximal Policy Optimization (PPO) for autonomous navigation. While successfully balancing and navigating on flat terrain, the robot encountered challenges in training the reinforcement learning model for uneven terrain navigation. Nevertheless, it remains operable on uneven terrain through manual keyboard control. Key contributions include creating a custom robot model and terrain and implementing a PID control strategy and PPO for navigation.

I. Introduction

Two-wheeled self-balancing robots present a challenging control problem due to their inherently unstable design. This project aimed to develop *BalanceBot*, a two-wheeled inverted pendulum robot, capable of navigating uneven terrain while maintaining stability. The objectives included implementing a PID controller for balancing, enabling manual navigation via keyboard commands, and utilizing reinforcement learning (RL) for autonomous navigation. The study addresses the complexities of stabilization and navigation, emphasizing the potential of RL for autonomous control.

II. Literature Survey

The design draws upon prior works in control systems and reinforcement learning. The PID controller is a well-established method for stabilization in robotics. Proximal Policy Optimization (PPO), an RL algorithm, was chosen over Deep Q-Networks (DQN) for its better convergence properties in continuous action spaces [1]. The project also references practical implementations such as backyard-robotics' [2] tutorial on balancing bots.

III. METHODOLOGY

A. Setup

The project initially began with ROS2 and Gazebo, which were later replaced by PyBullet and Gymnasium due to integration issues. PyBullet was chosen for its robust physics engine and ease of integration with Gymnasium, which is ideal for reinforcement learning tasks. A custom robot model was designed in XML format, comprising two wheels and a cuboidal body forming an inverted pendulum. The terrain was created using PyBullet's heightfield functionality, which allows for the generation of realistic uneven surfaces. The terrain features user-defined hills and valleys, providing a challenging environment for testing the robot's navigation capabilities. The following code snippet demonstrates how the terrain was generated in PyBullet:

```
terrain_shape = p.createCollisionShape(
    shapeType=p.GEOM_HEIGHTFIELD,
    meshScale=[0.05, 0.05, 2],  # Scale: x, y, height
    heightfieldTextureScaling=128,
    numHeightfieldRows=256,
    numHeightfieldColumns=256,
    heightfieldData=heightfield
)
```

The images below illustrate the custom two-wheeled robot model and the uneven terrain used for testing. Figure 1 shows the robot, which is designed to balance on two wheels, forming an inverted pendulum. This design is crucial for testing the robot's ability to maintain stability while navigating. Figure 2 depicts the uneven terrain, which includes user-defined hills and valleys. This challenging environment is used to test the robot's navigation capabilities, particularly its ability to handle complex surfaces while maintaining balance.

B. PID Controller

The PID controller was implemented to maintain the robot's balance by controlling its pitch and yaw, ensuring the robot stands upright. The controller operates in three modes: proportional, integral, and derivative, each contributing to the overall control strategy. The proportional mode (K_p) provides an immediate response to errors by applying a corrective force proportional to

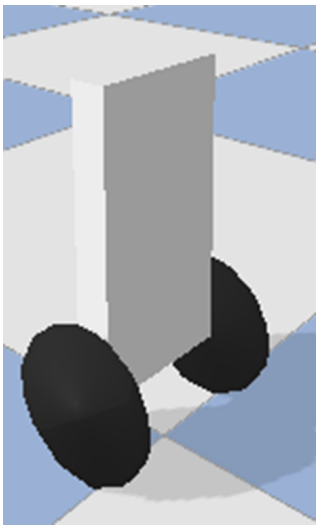

Fig. 1. The custom two-wheeled robot model.

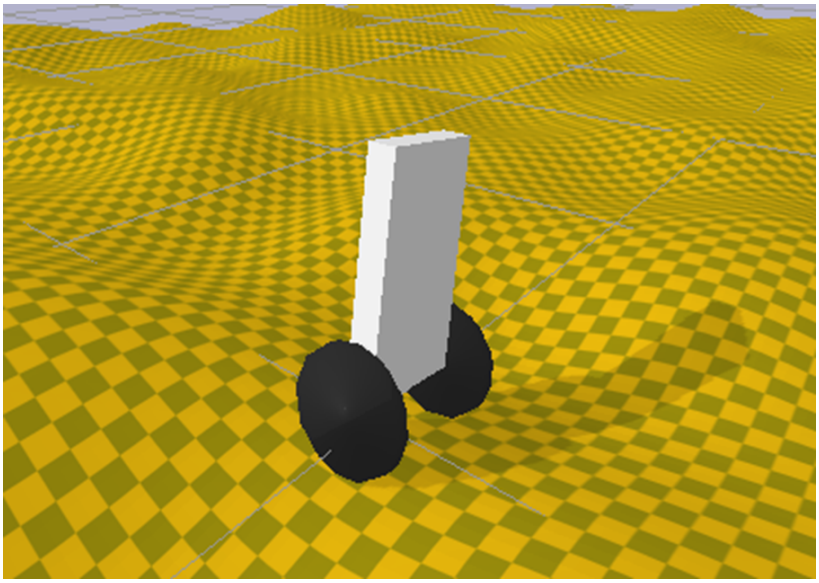

Fig. 2. The uneven terrain used for testing.

the current pitch error, helping the robot return to an upright position. The integral mode (K_i) addresses accumulated errors over time, ensuring that any persistent tilt is corrected by integrating past errors and applying a cumulative adjustment. The derivative mode (K_d) predicts future errors based on the rate of change of the pitch, allowing the controller to dampen oscillations and prevent overshooting by applying a force that counteracts rapid changes. The gains (K_p, K_d, K_i) were tuned iteratively, starting with K_p to achieve basic stability, followed by K_d and K_i to reduce oscillations and ensure smooth balancing. The following code snippet illustrates the implementation of the PID controller:

```
def pid_control(error, prev_error, integral, dt):
    Kp = 1.0  # Proportional gain
    Ki = 0.1  # Integral gain
    Kd = 0.05 # Derivative gain

    # Proportional term
    P = Kp * error

    # Integral term
```

```
integral += error * dt
I = Ki * integral

# Derivative term
derivative = (error - prev_error) / dt
D = Kd * derivative

# PID output
output = P + I + D
return output, integral
```

C. Keyboard Navigation

A manual navigation interface was implemented, allowing users to provide commands via the keyboard. These inputs were translated into force and yaw adjustments, enabling the robot to navigate both flat and uneven terrains.

D. Autonomous Navigation

Reinforcement learning was applied using PPO to enable the robot to autonomously reach predefined destinations. The RL agent controlled the robot's force and yaw. Although PPO outperformed DQN in initial tests, training the model to navigate uneven terrain remained challenging due to the increased complexity of the environment.

IV. Contribution

The key contributions of this project include:

- Development of a custom robot model and terrain in PyBullet.
- Implementation of a PID controller for real-time stabilization.
- Integration of PPO for autonomous navigation.
- Insights into the challenges of RL in complex terrains.

V. Results

The *BalanceBot* successfully balances and navigates flat terrain using both manual and autonomous control. On uneven terrain, manual keyboard navigation is functional, but the RL

model struggled to achieve reliable autonomous navigation due to insufficient training data and complexity of the environment. These findings highlight the need for further refinement of the RL training pipeline.

VI. Conclusion

This project demonstrates the feasibility of balancing and navigating a two-wheeled robot using PID and RL techniques. While the RL model faced limitations in uneven terrain, the robot's performance on flat surfaces and its manual operability on complex terrain underscore the potential for future improvements. Future work will focus on enhancing the RL model's robustness and exploring hybrid control strategies.

References

References

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

[2] Backyard Robotics, "Build a balancing bot with OpenAI Gym: Part I," [Online]. Available: https://backyardrobotics.com.