Intelligent Quadruped Robot Locomotion using PPO Agent

Priyanka Banka
Dept of EIE
Velagapudi Ramakrishna Siddhartha Engineering College
Vijayawada, India
priyankabanka7@gmail.com

Vijaya Lakshmi Korupu
Dept of EIE
Velagapudi Ramakrishna Siddhartha Engineering College
Vijayawada, India
vijaya.korupu@gmail.com

Chaitanya Varma Pakalapati
Dept of EIE
Velagapudi Ramakrishna Siddhartha Engineering College
Vijayawada, India
chaitanyavarma1257@gmail.com

Anusha Korukonda
Dept of EIE
Velagapudi Ramakrishna Siddhartha Engineering College
Vijayawada, India
anusha.korukonda03@gmail.com

Bindu Nalluri
Dept of EIE
Velagapudi Ramakrishna Siddhartha Engineering College
Vijayawada, India
bindunalluri9@gmail.com

***Abstract*—This project presents the application of the Proximal Policy Optimization (PPO) algorithm in Reinforcement Learning (RL) for autonomous locomotion in quadruped robots. These robots are increasingly used as transformative tools in industrial applications, such as transporting goods and conducting surveillance, especially in challenging or remote environments. The PPO agent interacts with the quadruped environment to learn an optimal control policy that enables the robot to navigate complex and varied terrains with stability and reliability. The agent takes states (observations) as inputs and outputs actions for which a delayed reward is received. The PPO algorithm optimizes policy updates within safe bounds, balancing exploration and exploitation to enhance adaptability and minimize instability in control tasks. This approach makes PPO well-suited for continuous control tasks essential to autonomous quadruped locomotion. The proposed approach is evaluated through simulations and real-time experimentation, with results showing significant improvements in stability, efficiency, and generalization.**

Keywords—autonomous, proximal policy optimization, locomotion, robot, surveillance, reinforcement learning.

I. INTRODUCTION

Quadruped robots, which have four legs that mimic animal-like locomotion, are increasingly used in various industrial applications, including surveillance, exploration, and logistics in remote or hazardous environments. These robots require sophisticated control strategies to navigate complex and often unpredictable terrains while maintaining stability and efficiency. Reinforcement learning, specifically PPO, has become a preferred approach to train quadruped robots due to its ability to learn and adapt autonomously in such environments. By interacting with the environment, a PPO-trained quadruped can adjust its actions based on feedback, thereby optimizing its control policy over time [1].

In recent years, Proximal Policy Optimization (PPO) has emerged as a highly regarded and effective reinforcement learning (RL) algorithm, particularly well-suited for tasks requiring continuous control in robotics. The Proximal Policy Optimization (PPO) algorithm improves training stability by gradually updating the policy, making small, controlled adjustments. Instead of allowing large, abrupt changes to the policy, PPO limits updates to stay within a safe range, reducing the risk of over-correcting. This approach helps maintain a balance between exploring new actions and reinforcing successful ones, which is essential in robotics, where environments can be unpredictable. By avoiding drastic changes, PPO helps robotic systems learn effectively in complex, uncertain conditions, making it popular for real-world robotic applications [2].

The safe policy update mechanism of the Proximal Policy Optimization (PPO) algorithm offers significant advantages for real-world implementations, particularly in preventing sudden or extreme policy adjustments that could lead to instability or even damage. This approach enables quadruped robots to perform smoother, more consistent movements across various terrains and effectively adapt to unexpected obstacles, thereby improving their ability to generalize across different environments. Studies have shown that with PPO, quadruped robots can learn stable and efficient gaits, allowing them to traverse challenging terrains with improved energy conservation a crucial factor for extended operations in industrial applications [3].

II. LITERATURE SURVEY

J. Hwangbo, I. Sa, R. Siegwart et al., proposed Deep Deterministic Policy Gradient (DDPG) for Agile Quadruped Maneuvers which used the DDPG algorithm to enhance a quadruped robot's agility and precision. By continuously controlling joint angles and torques, in this , DDPG enabled the robot to perform quick maneuvers like avoiding obstacles and sharp turns [4].

M. Haarnoja, A. Zhou et al., proposed Soft Actor-Critic (SAC) for efficient terrain adaptation. The SAC algorithm helped a quadruped robot move across rough and uneven terrains while conserving energy. SAC encouraged the robot to explore different movement strategies, finding energy-efficient ways to walk which allowed the robot to adapt to challenging surfaces without draining its battery quickly [5].

J. K. Ha and J. Schmidhuber proposed Multi-Agent Reinforcement Learning for Coordinated Quadruped Control, where each leg of the robot is controlled by a separate agent

using reinforcement learning. The agents work together to coordinate leg movements, helping the quadruped move smoothly over different terrains. This approach allows the robot to adjust each leg's action for better stability, especially when faced with disturbances [6].

Y. Nachum, D. Harb, and H. Xu proposed Hierarchical Reinforcement Learning for Quadruped Gait Adaptation which uses a two-layer system to adjust the robot's walking style. The top layer selects the appropriate gait based on the terrain, while the bottom layer controls leg movements. This setup enables the robot to adapt to different surfaces, improving stability and adaptability [7].

J. Schulman, S. Levine, et al., proposed the Trust Region Policy Optimization (TRPO) to achieve stable and efficient locomotion in quadruped robots. TRPO employs a trust region constraint to ensure controlled policy updates, minimizing the risk of destabilizing movements. This approach enables quadruped robots to navigate challenging and uneven terrains with enhanced stability and coordination, making it highly suitable for applications requiring reliable and adaptive robotic mobility [8].

III. Quadruped Locomotion Robot

Fig.1 shows the structure of a quadruped robot with labelled components. The robot has a main body called the Torso, which connects to four legs. Each leg has two key joints: the Hip joint, where the leg attaches to the torso, and the Knee joint, which allows further bending in the middle of the leg. These joints enable the robot's legs to move in a coordinated way, allowing it to walk or perform other locomotion tasks [9].

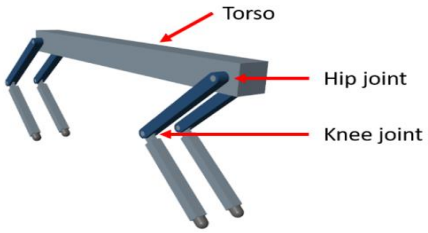

Fig. 1. Quadruped robot

In this project, a quadruped robot is controlled using a Proximal Policy Optimization (PPO) algorithm implemented in MATLAB's Reinforcement Learning (RL) toolbox. The robot receives 44 observations and performs 8 actions, which work as follows:

- Observations (44 in total): For each leg, 6 parameters are considered, including the hip and knee angles and velocities, as well as the foot-normal and friction forces, totaling 24 observations across 4 legs. For torso, 12 observations are considered. These include y (vertical) and z (lateral) positions of the torso center of mass (x (forward) is not considered) along with linear and angular velocities (vx,vy,vz, ωx, ωy, ωz) and Quanternion (orientation) qw,qx,qy and qz. Additionally, previous control torques for all joints are included to reflect the system's dynamic response and ensure smoother control actions. This comprehensive observation set provides essential information for the robot's motion control and stability in dynamic environments.
- Actions (8 in total): The robot has eight action values corresponding to the torques applied at each of its joint positions: two for each leg hip and knee joints (front-left, front-right, back-left, and back-right). These action values control the robot's movement by regulating the bending, extension, and propulsion of each leg, enabling coordinated locomotion. Together with the robot's observations, these actions help the agent optimize walking behaviour, ensuring stability and efficiency in various environments. Model parameters of Quadruped locomotion robot were shown in Table I [10].

TABLE I QUADRUPED ROBOT PARAMETERS

Parameter	Value
Distance b/w front and rear hip joints	L = 1 (m)
Length of the torso	L_back = 1.5 (m)
Link 1 length	l1 = 0.5517 (m)
Link 2 length	l2 = 0.5517 (m)
Desired height of the torso	h_final = 0.75 (m)
Torso mass	M = 2 (kg)
Leg link 1 mass	m1 = 0.2 (kg)
Leg link 2 mass	m2 = 0.2 (kg)
Initial body height	0.75 (m)
Initial foot displacement	0 (m)
Initial body speeds in x	0 (m/s)
Initial body speeds in y	0 (m/s)

IV. Steps in Implementation of PPO Algorithm for Quadruped Locomotion Robot

Proximal Policy Optimization (PPO) is a model-free, online, on-policy reinforcement learning algorithm. PPO aims to maximize the long-term reward by learning an optimal policy that balances exploration and exploitation while enforcing a stable and steady learning process. This is achieved by limiting the deviation between the current policy and the updated policy, often using a clipped surrogate objective function. The steps in implementing PPO learning algorithm for Quadruped Robot Locomotion were discussed here.

A. Set Up of Envinronment Interface

In simulating quadruped robot locomotion with a PPO agent, the robot model designed in Simscape Multibody is taken from Mathworks . Then, 44 observations and 8 actions required for the model are specified and reset function is used to initialize the robot's state by setting random or fixed initial conditions. It introduces random deviations in initial joint angles and angular velocities to simulate real-world uncertainties. The function also modifies parameters and helps the robot learn to handle various starting conditions.

B. Reward Function Shaping

After environment set up, now reward function is designed to train a robot to move efficiently and maintain balance. The reward function encourages the robot to maximize forward velocity, rewarding it for moving quickly while also penalizing it for falling. Additional rewards are given for maintaining the torso at a desired height and keeping it parallel to the ground, which promotes stability and control. Altogether, this shaping helps the robot learn to move forward quickly while staying upright and balanced.

C. Episode Termination

An episode in training or simulation ends if the robot loses stability in specific ways. Termination occurs if the torso's center of mass drops below 0.5 meters, or if the head, tail, or any knee joint touches the ground. Additionally, excessive tilting beyond set roll, pitch, or yaw angles (±0.1745 for roll and pitch, ±0.3491 for yaw) also stops the episode. These conditions ensure the robot resets when it becomes too unstable, reinforcing balance and control.

D. Building PPO Agent

The PPO agent is designed with separate actor and critic networks. The critic network approximates the value function, taking the observed state as input and outputting a single value representing the long-term reward. The actor network consists of three paths: a common path that processes the input state, a mean path to determine the average action, and a standard deviation path for exploration variability. These paths are connected within a layer Graph object, allowing for efficient backpropagation during training. Finally, the network structure is visualized using the plot function were shown in Fig.2 and Fig.3.

Fig. 2. Crictic function plot

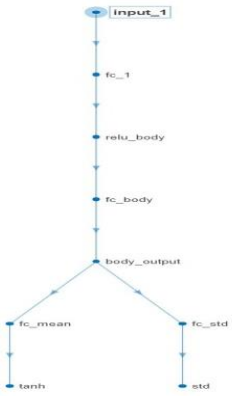

Fig. 3. Actor function plot

E. Training the PPO agent

Trained a PPO (Proximal Policy Optimization) agent which involves the agent in exploring an environment and collecting state-action-reward sequences, which are then used to update its policy. PPO utilizes a "clipped" objective function to ensure stable policy updates and prevent drastic changes. The algorithm balances exploration (trying new actions) with exploitation (using actions that yield high rewards). To encourage diverse exploration and avoid getting stuck in local optima, randomness, such as unique seeds or noise, is introduced into the process. This helps the agent continually improve its ability to maximize cumulative rewards.

After completing the training of the agent, it was tested and validated in a simulated environment to confirm stable walking behaviour. The process started by setting up a simulation that closely resembled the conditions used during training. The trained agent was loaded and run across multiple scenarios to evaluate key metrics such as stability, walking speed, and energy efficiency. The results were carefully analysed to ensure consistent, reliable performance. When performance issues arose, the reward function and policy parameters were adjusted to improve behaviour, and then the agent was re-tested. This iterative testing process is essential to ensure the agent is robust enough for real-world deployment.

V. PPO Agent Training for the Quadruped Locomotion Robot

The reinforcement learning environment is defined using a Simulink model, as illustrated in Fig.4. The system receives "observations" (state information from sensors) and calculates "rewards" (feedback on performance) based on actions taken by the robot. The central block represents the reinforcement learning agent, which processes the observation and chooses an "action" that controls the robot's movements. The process loops back, allowing the agent to learn and improve based on cumulative rewards over time, while checking if the task is completed. This setup helps the robot optimize its walking behaviour through trial and error.

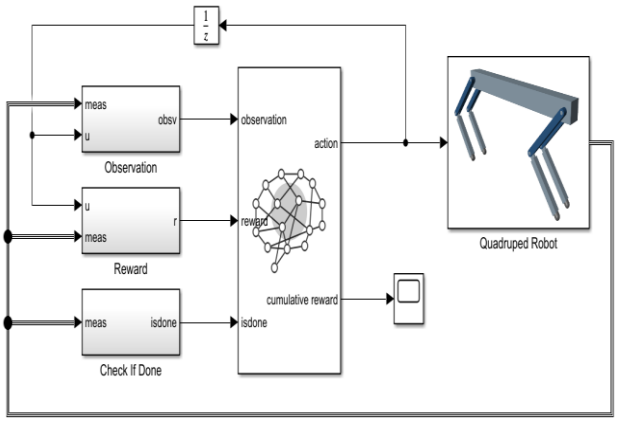

Fig. 4. RL Quadruped Robot environment

Training begins by fine-tuning parameters listed in Table II. These parameters play a crucial role in controlling how the agent learns and interacts with the environment [11]. For instance, the learning rate determines the speed at which the agent adapts to new experiences, while the clip factor ensures stable updates to its policy. The number of episodes defines how many training iterations are performed. Discount and GAE factors influence how the agent values future rewards and balance estimation errors, respectively. Maximum steps set a limit on the duration of each training session.

TABLE II PPO AGENT TRAINING PARAMETERS

Parameter	Value
Critic learning rate	1e-3
Actor learning rate	1e-4
Clip factor	0.2
No. of Epochs	3
Experience horizon	512
Discount factor	0.99
GAE factor	0.95
Entropy loss weight	0.01
Batch size	128
Max. Episodes	10000
Max. steps per episode	400
Simulation time	10 (s)
Sample time	0.025 (s)

The reward function is crucial for guiding a quadruped robot towards stable and efficient locomotion by rewarding desired behaviors such as maintaining balance and minimizing energy consumption. It helps the robot adapt to dynamic environments, ensuring it learns to perform tasks like walking steadily and avoiding undesirable actions like falling. Fig.5 shows reward function for Quadruped Robot.

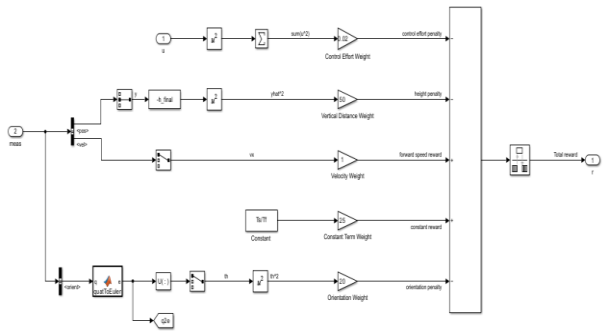

Fig. 5. Quaruped Robot reward function

The training progress shown in the Fig.6 illustrates the performance of robot in episode manager .

Fig. 6. RL agent training progress

After further training, we observed that the robot has started to move, but it still does not achieve the expected walking behaviour. Although adjustments to the reward function and additional training have led to some initial movement, the robot showed improved performance.

VI. RESULTS

Fig.7 illustrates the robot's velocity changes over time, reflecting its movement dynamics during the training process. The PPO agent is effectively controlling the robot's forward motion while keeping lateral and vertical movements minimal, helping it remain balanced and stable.

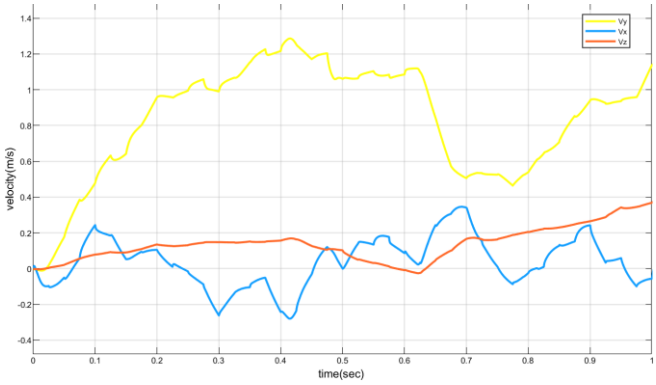

Fig. 7. Quadruped robot velocity

The quadruped robot orientation is shown in Fig. 8. The plot demonstrates that the PPO agent is effectively controlling the robot, making small adjustments in orientation to maintain balance and stability.

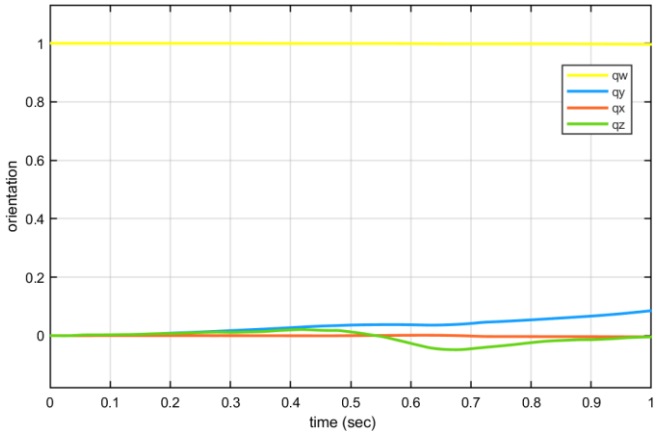

Fig. 8. Quadruped robot orientation

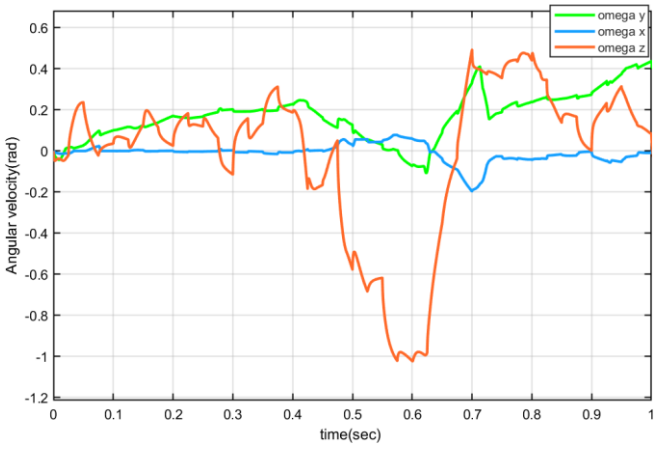

Fig. 9. Quadruped robot angular velocity

Fig. 9 shows the angular velocities (in radians per second) for the three axes, over a period of time from 0 to 1 second.

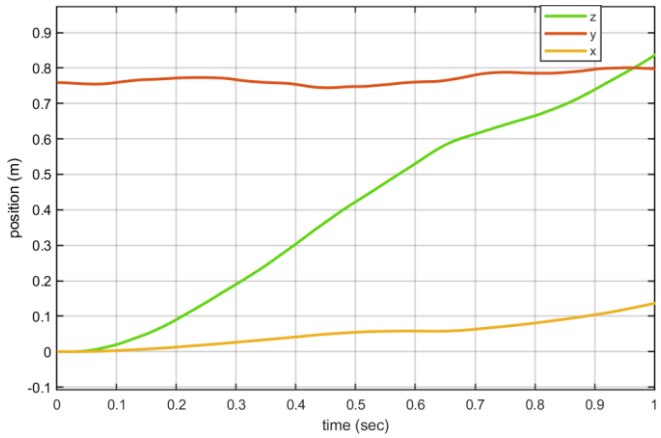

Fig. 10. Response of robot w.r.t position

Fig.10 shows the robot's position changed over time in three directions: x, y, and z.

VII. Conclusion and Future Scope

This project successfully demonstrates the application of the Proximal Policy Optimization (PPO) algorithm in training a quadruped robot for autonomous locomotion. By utilizing PPO's stability in policy updates, the robot learned to navigate efficiently, ensuring balance. Future work on this project could focus on improving the robot's energy efficiency, enhancing real-world testing in diverse environments, and developing multi-robot coordination for collaborative tasks.

References

[1] Y. Wang, R. Sagawa, and Y. Yoshiyasu, "Learning Advanced Locomotion for Quadrupedal Robots: A Distributed Multi-Agent Reinforcement Learning Framework with Riemannian Motion Policies," *Robotics*, vol. 13, no. 6, p. 86, May 2024. doi: 10.3390/robotics13060086.

[2] S. Gai, S. Lyu, H. Zhang, and D. Wang, "Continual Reinforcement Learning for Quadruped Robot Locomotion," *Entropy*, vol. 26, no. 1, p. 93, Jan. 2024. doi: 10.3390/e26010093.

[3] Y. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal Policy Optimization Algorithms," *arXiv preprint*, doi:1707.06347, 2017.

[4] J. Hwangbo, I. Sa, R. Siegwart, and M. Hutter, "Control of a Quadrotor with Reinforcement Learning," *IEEE Robotics and Automation Letters*, vol. 4, no. 4, pp. 667-674, Oct. 2019. doi: 10.1109/LRA.2019.291963

[5] M. Haarnoja, A. Zhou, et al., "Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor," in *Proceedings of the 35th International Conference on Machine Learning (ICML)*, Stockholm, Sweden, 2018, pp. 1861–1870.

[6] J. K. Ha and J. Schmidhuber, "World models," *Proc. NeurIPS*, 2018, pp. 1-10. doi: 10.5555/3327757.3327805.

[7] Y. Nachum, D. Harb, and H. Xu, "Data-Efficient Hierarchical Reinforcement Learning," *Advances in Neural Information Processing Systems (NeurIPS)*, pp. 3303–3313, 2018. [Online]. Available: https://arxiv.org/abs/1805.07733

[8] J. Schulman, S. Levine, P. Abbeel, M. Jordan, and P. Moritz, "Trust Region Policy Optimization," in *Proceedings of the 32nd International Conference on Machine Learning (ICML)*, Lille, France, 2015, pp. 1889-1897. doi: 10.5555/3045118.3045361.

[9] Di Carlo, J., Wensing, P. M., & Kim, S. (2018). "Dynamic locomotion in the MIT Cheetah 3 through convex model-predictive control," *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Madrid, Spain, pp. 1-8. doi: 10.1109/IROS.2018.8593410

[10] MathWorks. (n.d.). *Quadruped Robot Locomotion Using DDPG Agent*. MATLAB and Simulink Examples. Retrieved November 10, 2024,fromhttps://www.mathworks.com/help/reinforcement-learning/quadpud-robot-locomotion-using-ppo-agent.html.

[11] Heess, Nicolas, Dhruva TB, Srinivasan Sriram, Jay Lemmon, Josh Merel, Greg Wayne, Yuval Tassa, et al. 'Emergence of Locomotion Behaviours in Rich Environments'. *ArXiv:1707.02286 [Cs]*, 10 July 2017. https://arxiv.org/abs/1707.02286.

[12] Lillicrap, Timothy P., Jonathan J. Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. 'Continuous Control with Deep Reinforcement Learning'. *ArXiv:1509.02971 [Cs, Stat]*, 5 July 2019. https://arxiv.org/abs/1509.02971.