Locomotion of Hexapods using Reinforcement Learning

Abhigyan Mehrotra: U20210002
Prashant Mishra: U20220067
Moksh Soni: U20220101
Vaibhav Chopra: U20220092

May 2025

Abstract

This project demonstrates experiments applying Tabular Q-Learning, DQN, and Proximal Policy Optimization (PPO) to solve the locomotion problem of hexapods. Designing algorithms for hexapods, or legged robots in general, is challenging due to the large state space and sparse rewards. Through discretization of the action and state spaces, we demonstrate that Q-Learning can be used to learn the behavior of "staying alive." Furthermore, we implemented PPO to eliminate over-discrete action spaces and learn both "staying alive" and walking toward the goal. Our experiments with DQN showed insignificant learning. PyBullet and a pre-existing URDF file were used for simulation purposes [**?**].

Background, Problem Statement, and RL Formulation

Reinforcement Learning was first applied to hexapods in the 1990s. Due to computational limitations, interesting solutions emerged to learn the required behavior. *Kirchner* attempted to learn elementary leg movements using hierarchical Q-Learning [**?**]. *Zennir Youcef* and *Couturier Pierre* used a distributed version of Q-Learning to achieve locomotion [**?**]. As the computational power of microcontrollers and general-purpose computers increased, non-tabular methods and models with higher-dimensional state spaces were implemented. *Mohammadali Shahriari* implemented a Q-Learning algorithm with a fuzzy reward structure to guide the robot toward goal states [**?**]. *Huiqiao Fu et al.* implemented a deep reinforcement learning-based motion planner to learn feasible foothold and center of mass trajectories [**?**].

Objective of the Project

Our objective was to develop a reinforcement learning-based control system that stabilizes the hexapod and moves it toward a desired goal, both in simulation and on hardware.

RL Formulation

To develop an appropriate policy for the hexapod, it is necessary to choose the right RL representation. The most general Markov Decision Process (MDP) includes all possible actions and states. For a six-legged robot with three joints per leg, the state and action spaces are not only large but also continuous. Specifically, our simulated robot has six legs and three joints per leg, with each joint rotating between $+135^\circ$ and -135°. Discretizing into 10° intervals yields 27 actions per joint. Therefore, at each timestep, the robot can choose from 27^{18} actions. Similarly, analyzing the state space results in hundreds of states, tracking each joint's position, velocity, forces, and moments, along with the body's position, velocity, and orientation. Altogether, the total number of state variables is $10 * 18 + 7 = 187$.

Given the large number of state-action pairs, several MDP formulations are possible:

1. Divide the problem hierarchically, learning behaviors for each sub-level.
2. Use function approximation to eliminate discrete states and actions.
3. Treat each leg as a separate agent, creating identical MDPs per leg.

We chose a basic approach:

- At every timestep, the robot selects actions for all motors.
- The robot observes new states and rewards for each joint and the body.
- Rewards and new states are used directly, rather than passed through hierarchical or distributed structures.

The reward structure used to achieve locomotion toward the goal is:

$$R_t = w_1 D_t - w_2 |H_t| - w_3 E_t - w_4 T_t$$

where w_1, w_2, w_3, w_4 are weights, and:

- D_t is the distance from the goal.
- H_t is the difference between actual and desired height.
- E_t is energy consumption.
- T_t is the sum of roll and pitch.

Methodology and Contributions

We experimented with three primary methods:

1. Q-Learning
2. DQN
3. Proximal Policy Optimization (PPO)

Q-Learning Experiments

1. **Vanilla Q-Learning**: Tabular Q-Learning requires discretizing state and action spaces. As a baseline, we used a simple approach: each motor had two actions (move to $+135°$ or $-135°$), and states included only the robot's height, roll, pitch, and yaw—yielding four states and 2^{18} actions. After 10,000 episodes, almost no learning was observed.
2. **Synchronized Q-Learning**: We grouped random legs into sets of three. Each motor retained two actions ($+135°$ and $-135°$), reducing the action space to 2^6. If three actions per motor were allowed, the action space became 3^6, still manageable. Meaningful behaviors were learned for "staying alive" under both configurations.

Serial_No	Number of Actions	Number of States	Synchronized	Q-Table Size
1	2^{18}	4	No	4×2^{18}
2	2^6	4	Yes	256
3	3^6	4	Yes	2,916

Table 1: Q-Table configurations

The Q-Learning reward function for "staying alive" was simplified by removing D_t and E_t terms:

$$R_t = -w_2 |H_t| - w_4 T_t$$

DQN

We implemented DQN using a discretized action space similar to synchronized Q-Learning:

- 2 hidden layers
- Discretized action space
- ReLU activation

Proximal Policy Optimization

PPO enabled using continuous action spaces. We implemented PPO (using *stable_baselines3*) for both "staying alive" and reaching a goal.

Results

Q-Learning Results

Summary of Q-Learning experiments:

Sr No	Synchronized	Actions	States	Q-Table Size	Learned?	w_2	w_4	Episodes
1	No	2^{18}	4	4×2^{18}	No	0.1	0.1	10,000
2	Yes	2^6	4	256	Yes	2	0.1	4,000
3	Yes	3^6	4	4,374	Yes	2	0.1	10,000
4	Yes	2^6	4	256	Yes	2	0.1	10,000
5	Yes	3^6	4	4,374	No	2	0.2	1,000
6	Yes	3^6	4	4,374	No	0.1	0.1	10,000
7	Yes	3^6	4	4,374	Yes	2	0.1	10,000
8	Yes	2^6	4	256	Yes	2	0.1	5,000
9	Yes	3^6	4	4,374	No	2	0.2	50,000
10	Yes	2^6	4	256	No	1	0.7	7,000

Table 2: Summary of Q-Learning experiments

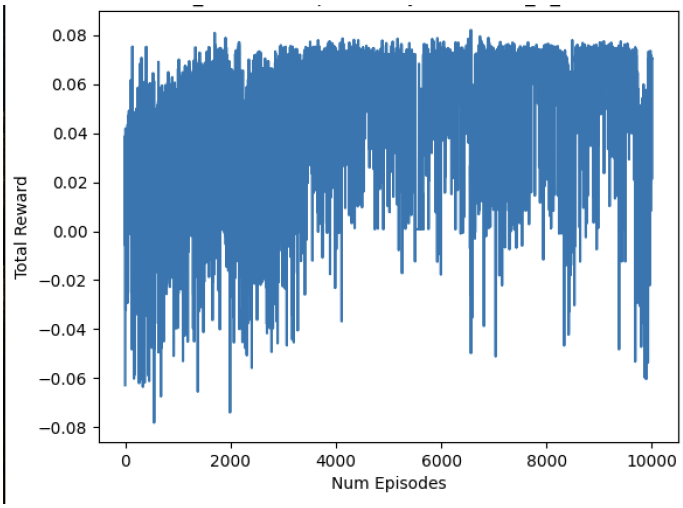

Figure 1: Total Rewards per Episode – Case 1 (No learning)

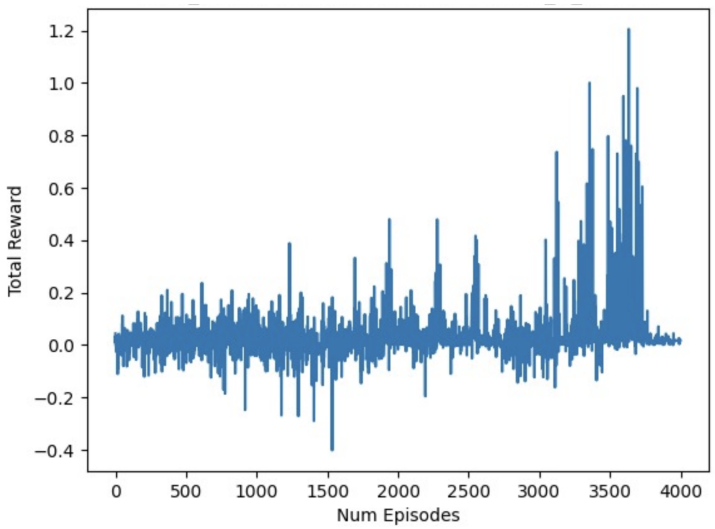

Figure 2: Total Rewards per Episode – Case 2 (Learning achieved)

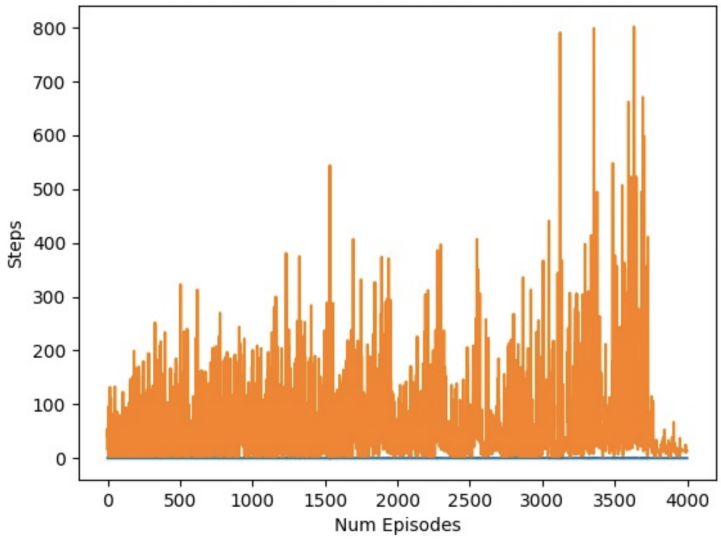

Figure 3: Survival Steps per Episode – Case 2

DQN

Experiments with DQN did not yield meaningful learning and are not detailed here.

PPO

PPO produced the most promising results. Despite negative rewards, the robot survived longer:

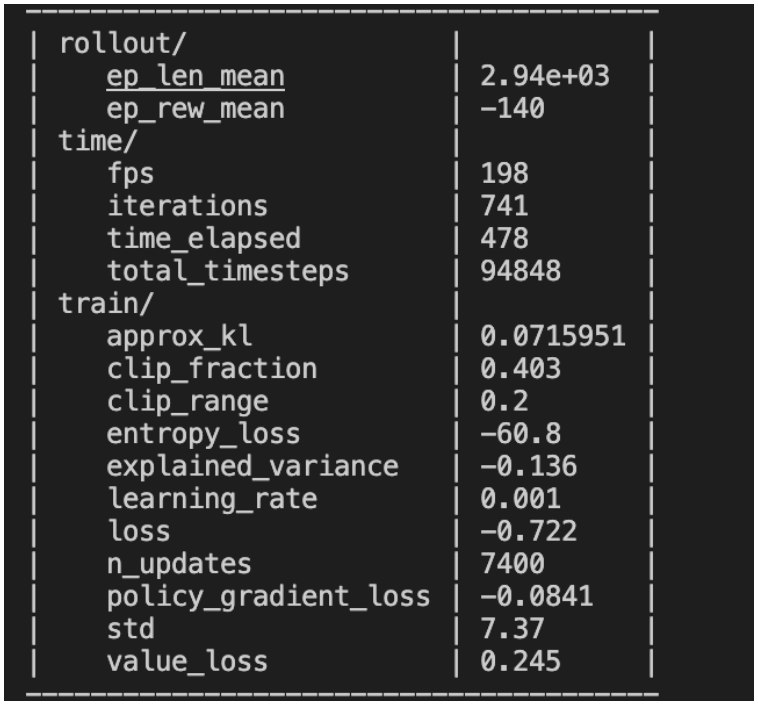

Figure 4: Training Run at the End of PPO Training

Future Work and Hardware

In future work, we plan to implement this algorithm on the following hardware:

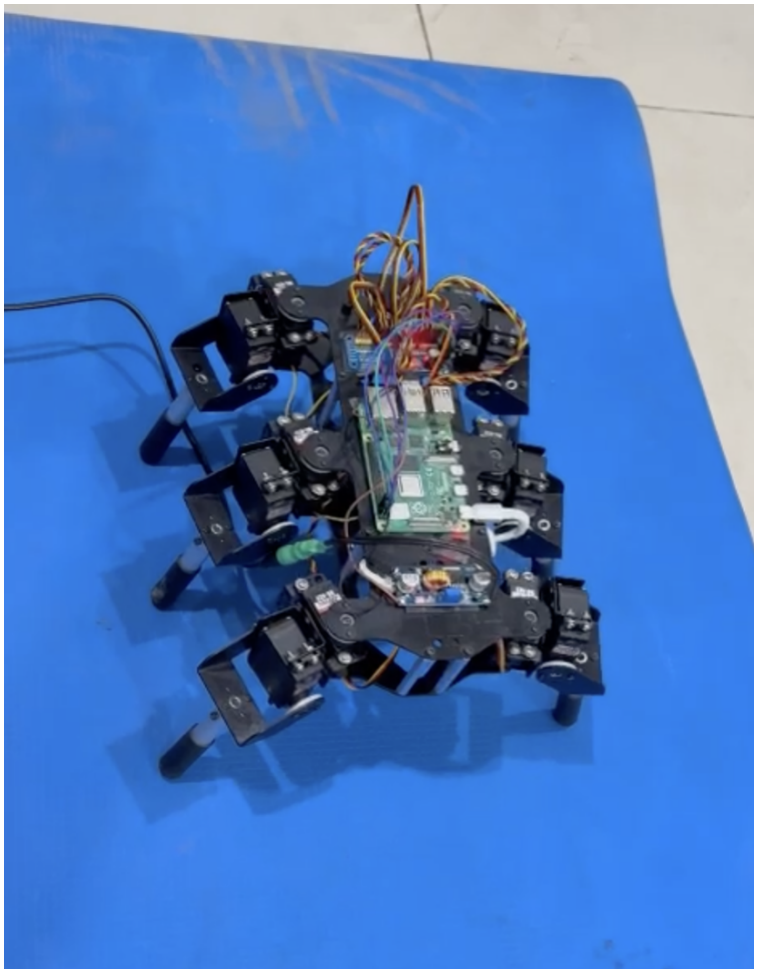

Figure 5: Target Hardware Platform

Citations

1. Kirchner, F. (1998). Q-learning of complex behaviours on a six-legged walking machine. *Robotics and Autonomous Systems*, 25(3–4), 253–262. https://doi.org/10.1016/S0921-8890(98)00049-2

2. Youcef, Z., & Couturier, P. Control of the trajectory of a hexapod robot based on distributed Q-learning. Centre de recherche LGI2P, EMA-Site EERIE, Parc scientifique Georges Besse, Nîmes, France.

3. Shahriari, M., & Khayyat, A. A. (2013). Gait analysis of a six-legged walking robot using fuzzy reward reinforcement learning. In *Proceedings of the 13th Iranian Conference on Fuzzy Systems (IFSC)* (pp. 1–4).

4. Fu, H., & Tang, K. (2025). A deep reinforcement learning-based motion planner for hexapod robots integrating a trajectory optimization-based transition feasibility model. [Manuscript in preparation / conference paper].

5. https://github.com/Ammar500Issa/Hexapod/tree/main