J. Integr. Sci. Technol. 2023, 11(4), 574 Article

Journal of Integrated
SCIENCE & TECHNOLOGY

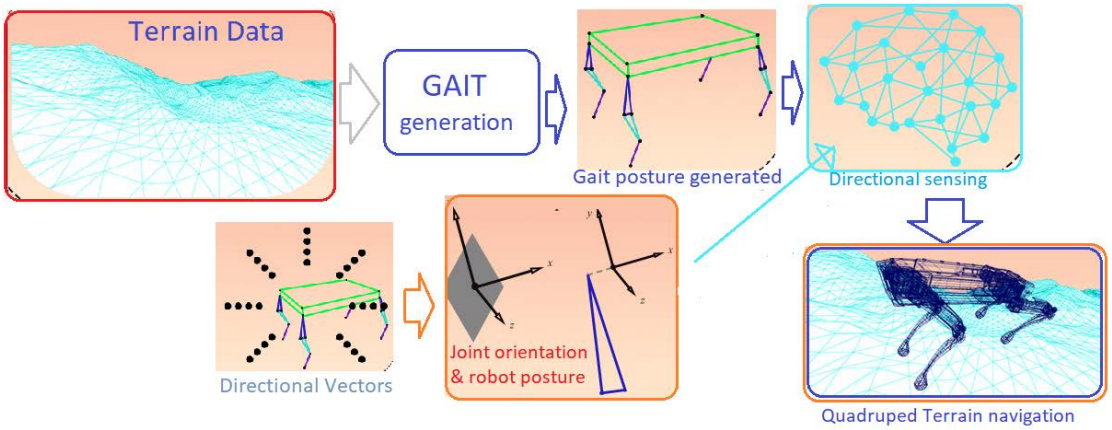
Using TRPO to control quadruped gait behaviors

Aditya Chhabiraj Jaiswal, Shweta Sinha, Priyanka Makkar

Amity School of Engineering and Technology, Amity University Haryana, India

Received on: 18-May-2023, Accepted and Published on: 17-Jul-2023

ABSTRACT



Quadruped robot locomotion control is tough and complex due to the redundant DOF and interlocked movement of their four legs, even though a suitable control method has a significant impact on the performance of the control. The following contributions are made by the paper to the creation of the ideal gait controller for the legged robot. The quadruped robot's fundamental mechanical parts are first recreated in a virtual setting. Second, a TRPO model based on KL divergence is created, and the model's accuracy and computation speed are evaluated. Using curriculum learning and actor-critic approaches, the best gaits for various walking tasks are discovered. Finally, the virtual model is updated to incorporate the learnt gaits together with many additional behaviors, including vision and directional variance. According to preliminary findings, the robot can efficiently navigate and correct its walking paths in real time with no processing overhead.

Keywords: Control system, Deep reinforcement learning, Gait control, KL divergence, Quadruped robot, Trust region policy optimization

INTRODUCTION

Robots with several legs have been utilized extensively to explore different terrains. Quadruped robots have the advantage of having balanced properties for structural complexity, mobility, and locomotive stability among all extant categories of legged robots.[1,2] For such complex control situations, numerous control methods and analysis control theory are typically used. Deep reinforcement learning (DRL), a new alternative technique for enhancing leg motor skills, has lately come into existence due to the quick development of the artificial intelligence area. DRL's main tenet is that the control policy learns to choose actions based on the reward it receives from the environment in order to maximize benefit.[5] DRL has been used to automate elements of the design process, simplify the design of locomotion controllers, and learn behaviors that earlier control techniques were unable to accomplish.[6] In recent years, a lot of effort has been paid to the study of DRL algorithms for legged robots.

The model presented in this research is based on deep reinforcement learning (DRL) and artificial neural network (ANN), and it effectively replaces the conventional analysis-based control theory, including inverse kinematics, differential equations of motion, and governing equations. The ANN model is trained using training data produced by DRL, and it is then utilized to operate a quadruped robot. A virtual replica of a typical quadruped robot is used to test the experimental validity of the proposed AI-based robot-control system. The outcomes demonstrate that the suggested approach is a potential new controller that can take the place of the mathematically puzzling robot control system.

RELATED WORK

To transfer the rotating forces from electric motors to the robot leg joints for motion generation, the majority of the reported quadruped robots use rigid components such as beams and

Corresponding Author: Aditya Chhabiraj Jaiswal, Shweta Sinha, Amity School of Engineering and Technology, Amity University Haryana, India. Tel: +919004768230, +919868140118
Email: a.jaiswal151000@gmail.com, ssinha@ggn.amity.edu

Cite as: *J. Integr. Sci. Technol., 2023, 11(4), 574.*
URN:NBN:sciencein.jist.2023.v11.574

 ISSN: 2321-4635
http://pubs.thesciencein.org/jist

bearings. These robots benefit from quick responses and precise movements, but their motion ranges are constrained by the difficulty of maintaining steady mobility.

The mass center criteria, which asserts that a robot is statically stable if the projection of its center of mass CoM onto a horizontal plane, falls within the support grid, is one of the most well-known statically stable gait stability tests. To track the planned trajectories of walking robots, a number of control techniques, including Model-Based Algorithm (MBA) based on the feedback linearization approach, robust control, and others, have been developed and put into use.

The complexity of conventional control methods is gradually raised when more scenarios are taken into account in order to explore quadrupedal locomotion on uneven terrain in greater detail. As a result, the accompanying development and maintenance become labor and time-intensive, and they continue to be susceptible to extreme circumstances.

Deep reinforcement learning (DRL), a new alternative technique for enhancing leg motor skills, has lately come into existence due to the quick development of the artificial intelligence area. The fundamental tenet of DRL is that the control policy learns to choose actions that will yield the most benefit based on the reward from the environment.[5] DRL has been used to automate some of the design process, learn behaviors that earlier control methods were unable to achieve, and simplify the design of locomotion controllers.[6,9] In recent years, a lot of effort has been paid to the study of DRL algorithms for legged robots.

Robotic quadrupeds typically use pipelined locomotion control methods, such as state estimation, contact scheduling, foot placement planning, etc.[5,7] Both expert knowledge and precise dynamic models of the quadruped robot are needed for these activities. Model-free Reinforcement Learning (RL) techniques, on the other hand, don't need to be familiar with the robot system beforehand. In order to better the motion controller and achieve the motion target specified by human operators, RL algorithms can interact with the robot motion environment physically or virtually.

There has been a lot of scientific interest in the creation of new algorithms that can determine a target's position more accurately while also taking eyesight and proprioceptive states into account. The authors of [8] proposed a regularized location estimator (RLE) open form technique for the adaptive control system. This method uses maximum likelihood estimation. They determined that on a desktop computer with 4 GB RAM and a 2.1 GHz processor, calculating a position only required 8 microseconds. When the classic localization problem was tackled using a regularized approach, the authors of [6] found that 0.1 (the value of the regularization parameter) was sufficient to get good results. And should be lowered as well if the degree of poor conditioning is decreased. In order to maximize coverage for a given accuracy, the authors of[10] used a genetic algorithm; nevertheless, they failed to assess the impact on pattern reconstruction.

One of the big challenges in reinforcement learning is the amount of simulation that needs to be performed for the agent to converge to a solution. All state-action pairs must be visited many times before the solution converges. As the state space and action space increases the amount of simulation required also increases. To use continuous state and action spaces and allow for faster convergence faster convergence an approximation of the policy function can be used, or so-called policy gradient method. Different types of approximations can be used, but in this research each function is approximated using a neural network that maps the respective input to the respective output.

Modern, cutting-edge locomotion controllers frequently use a pipelined control strategy. The MIT Cheetah[27] utilises a state machine over contact circumstances, creates straightforward reference trajectories, use model predictive control[9] to prepare for desired contact forces, and then implements Jacobian transpose control to make those forces happen. The ANYmal robot[23] uses the inverted pendulum model[13] for foothold planning, CMA-ES[19] for parameterized controller optimisation, and a hierarchical operational space control problem[22] for body motion and joint torques. Although these techniques can result in efficient gaits, they necessitate extensive prior knowledge of the locomotion problem and, more significantly, of the dynamics of the robot. The proposed approach, in contrast, tries to control the robot without having any prior knowledge of either the dynamics or the gait. Since all learning occurs purely through real-world interaction, there are no assumptions about having access to any trajectory design, foothold planner, or robot dynamics model.

Deep RL has been widely utilised to learn locomotion policies in simulation[4] and even transfer them to real-world robots.[21] However, this unavoidably results in a performance loss due to differences in the simulation and necessitates proper system identification. It has been difficult to apply such algorithms directly in the real world. Low-dimensional gait parameterizations[8] or basic, inherently stable robots[14] are frequently used in real-world applications, or both. In contrast, it is demonstrated how neural-net rules can be used to directly learn locomotion abilities in the actual environment.

The proposed approach, maximises the weighted sum of the expected return and the expected entropy of the policy, is based on trust region policy optimisation. Similar approach has been applied in a variety of situations, including optimum control and inverse RL.[5] Maximum entropy RL has the benefit of producing policies that are somewhat robust, as the introduction of structured noise during training encourages the policy to explore a larger portion of the state space and increases the resilience of the policy.[16] However, the entropy term's weight is often determined arbitrarily.[15]

Based on the background research conducted, it is observed that, this value is extremely sensitive, and manual adjusting may make it challenging to apply the maximum entropy framework in actual situations. Instead, limiting the predicted entropy of the policy and automatically adjusting the temperature to meet the limitation is suggested. Constrained MDP is a concept that is used in the formulation and was recently studied in [6]. In contrast to the situation, where the constraint clearly depends on the policy, these works only take into account constraints that depend on the policy indirectly through the sampling distribution. The method applies directly to the entropy of the present policy but is also strongly related to KL-divergence requirements that restrict the policy change between iterations.[2] On both simulated benchmarks and observed data, it was found that this straightforward tweak significantly decreases the burden of parameter tuning.

PROBLEM DOMAIN

Before conceiving a gait controller, it is essential to first comprehend the characteristics of the old techniques, especially when taking adaptability and dynamic computation limits into account. The absolute continuity of movement that the system must be able to process without delays presents the biggest obstacle in creating a gait controller.

Other challenges with these characteristics include:

- Computational Convergence: Before the solution converges, all state-action pairs must be visited numerous times. The quantity of simulation required rises as the state space and action space expand. The so-called policy gradient approach, an approximation of the policy function, can be utilized to leverage continuous state- and action-spaces and enable faster convergence.
- Synchronous control and Adaptability: To design an effective gait pattern, the controller must develop a pertinent motor control pattern that can support synchronous joint movement under a variety of circumstances.
- Proprioceptive and Visual States: While controllers based on either of the aforementioned paradigms exhibit relatively few diffractions, a special type of parameterized vector control must be used for a controller to fully exploit both proprioceptive and visual states.

METHODOLOGY

As soon as we have a thorough understanding of the project that has been suggested, we shall talk about the development technique.

Kinematical Analysis

A robot that is bio-inspired is the quadruped robot. A redundant degree-of-freedom (DOF) system is a good example of a robot with four legs. A quadruped robot has a stiff body and four legs. The degrees of freedom for each leg are 3.[15] By means of revolute joints, each link is joined to every other link. The coxa joint, femur joint, and tibia joint are the three joints.

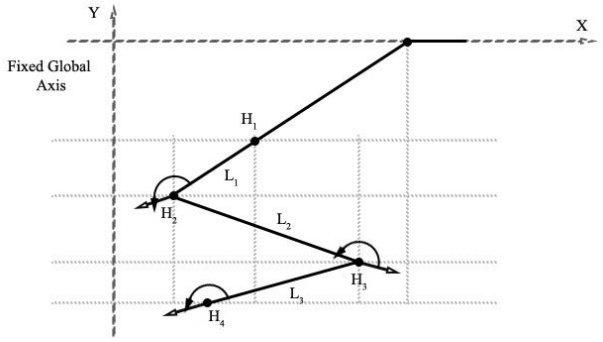


Figure 1. Quadruped robot single-leg coordinate systems. It consists of 3 links and 3 revolute joints, which are L1, L2, L3, and J1, J2, J3.

A quadruped robot may move in a variety of animalistic ways, including walking, jogging, pacing, cantering, galloping, creeping, and trotting. We must determine the forward and inverse kinematics of a quadruped robot in order to modify its gait.

Degrees of freedom (DOFs) in the robot body are split into two groups: the major DOFs, which are necessary for walking, and preserving the secondary DOFs. By finding a proper kinematic

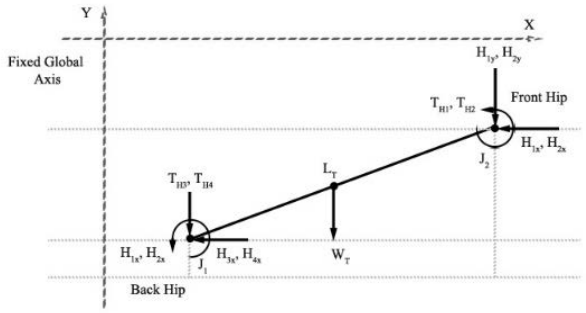


Figure 2. Kinematical Analysis of the body link separated from the mechanical legs

resolution of the motions of the minor DOFs, the motions of the major DOFs can be realized.[11] Inverse kinematics problems can be solved using a variety of methods, including the Gradient Projection method (GP), Weighted Least Norm method (WLN), and Extended Jacobian matrix method (EJ method). Inverse kinematics are solved using the extended Jacobian matrix (EJ) approach, and the robot is controlled using the Time-Pose control method. An improved Jacobian pseudoinverse (mIJP) approach that has been developed was also used to overcome the inverse kinematics issue. Gait planning for a robot based on an SpotMini was created utilizing inverse kinematics and the Jacobian of the full body.

Virtual Replication

We developed a DH representation of the robot's structure for simplicity in the structural replication process based on our thorough examination of the SpotMini robot. The body of SpotMini, which contains computers and cameras, and its four legs are its main body sections. Each leg has a ball joint at the hip, where the upper leg attaches to the torso, as well as a hinged knee that joins the upper and lower leg pieces.

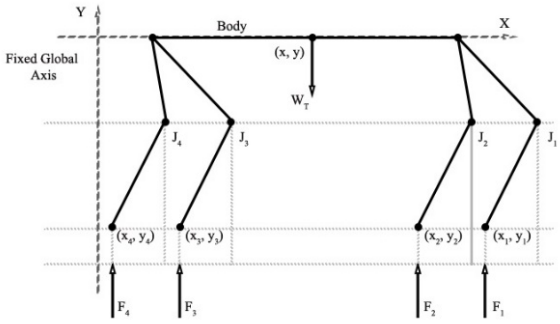


Figure 3. DH representation for a Quadruped robot in mammal configuration

In Autodesk Maya 2023, a virtual replication was made in accordance with the DH representation. The virtual model has three revolute joints in each leg, allowing for three degrees of freedom of movement.

The quadruped is broken in two separate submeshes – one for torso/body and another for set of mechanical legs with joints and actuations. The robot's legs were eventually attached to the static body's mesh after the torso was first built.

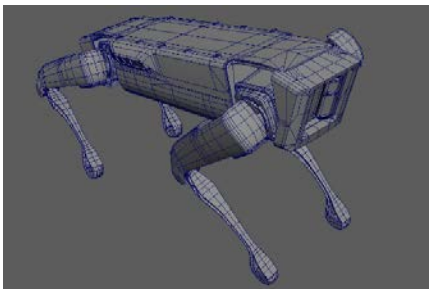


Figure 4a. Virtual model of Quadruped robot

The model had 46,830 triangles and 31,480 vertices after final design. Its measurements are 90cm x 40cm x 70cm, which are identical to the actual SpotMini, and a Unity rigidbody component with a mass of 30 kg was later added. The mechanical legs were created based on the kinematical and structural analysis of a real-world mechanical leg with all the accuracies in joints and actuation functionalities. Each leg consisted around 5,110 vertices and 5,010 traingles. Each leg acts as a complete dynamic surface with various movable parts and components to it.

Perception

For the agents to be able to observe and record their environment, we need to add some sort of sensors to them which can help them to detect and determine their environment. To accomplish this, we utilize the Raycast Perception Sensor provided by Unity Engine.

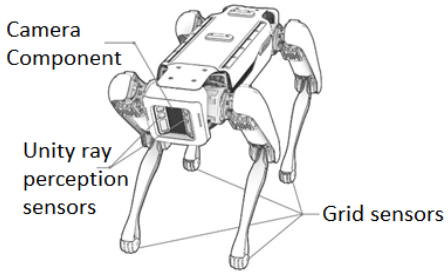


Figure 4b. Perception setup with various sensors placement on virtual Quadruped robot model

The observation vector is made up of all the environmental factors the AI system keeps track of during training.

A ray-casting system was implemented originating from the quadruped's model camera-head. It has an infinite reach and is intended to find any terrain-designated object in the scene. If the ray-casting system hits any object, the ray in question turns red or are white otherwise.

We provide the quadruped with 3 sets of ray-cast sensors each having three child ray-casts. These 3 sets of ray-casts are extended in different directions each – the middle one directed towards the forward vector of robot while the other two also directed towards the forward vector but with an angle deviation of +15 degrees and -15 degrees in horizontal axes. This helps the agent to detect any sort of obstacle in its path – whether forward, left, or right.

A Unity scene represents objects in a three-dimensional space. Unity must take a view and "flatten" it for display since the viewer's screen is two-dimensional. It does this using cameras. So, to render our simulation, we create a camera by adding a Camera component to our quadruped. The module defines a follow camera rig system which facilitates the target following based on a provided offset. The minimal offset can be defined as both in x-z and x-y quadruped facilitating a cam follow configuration.

Basic Architecture

Reinforcement learning (RL) problems are frequently used to model quadrupedal movement in the context of Markov decision processes. According to the fundamental concept, policies with the highest rise in incentives should be followed.[30] For curved areas, the first-order optimizer is not very precise. Learning a control strategy that enables a legged robot to maximize its predicted return on a given task is the goal. At each time step, the robot observes a state in the surrounding area and determines a course of action based on its policy.[21]

$$\max_{\theta} J(\pi_\theta) = \mathop{\mathrm{E}}_{r \sim \pi_\theta}[\sum_{t=0}^{\infty} \gamma^t r_t]$$

Equation 1. Standard mathematical representation of Trust region policy optimization

Following this action, the robot experiences a novel state and receives a scalar reward. As a result, a trajectory can be created by using this contact process repeatedly. Formally speaking, to solve the RL problem, the robot must develop a decision-making strategy that maximizes the projected discounted return.

In RL, a policy is optimized for the highest possible expected discounted rewards. Nevertheless, PG performance is hampered by a few issues. To update the policy in that direction, PG computes the rewards' steepest ascending direction (the policy gradient g). Because the learning rate is insensitive to the problem's geography, PG has a severe convergence problem. The parameter changes that are influenced by the terrain are limited by TRPO. But it is not straightforward to offer this answer. We modify the policy using basic model parameters.

$$g = \nabla_\theta J(\pi_\theta) = \mathop{\mathrm{E}}_{r \sim \pi_\theta}[\sum_{t=0}^{\infty} \gamma^t \nabla_\theta log \pi_\theta(a_t|s_t) A^{\pi_\theta}(s_t, a_t)]$$

$$\theta_{k+1} = \theta_k + \alpha g$$

Equation 2. Mathematical representation of Steepest ascent policy gradient

Finally, given only one policy update, we sample the entire trajectory. We are unable to change the policy at every interval. But after some time of testing, significant roadblocks appeared. The technical difficulties with PG can be summed up as follows:

- Large policy changes disrupt training,
- Unable to map changes between policy and parameter space,
- Incorrect learning rate results in disappearing or exploding gradient,
- Poor sample efficiency.

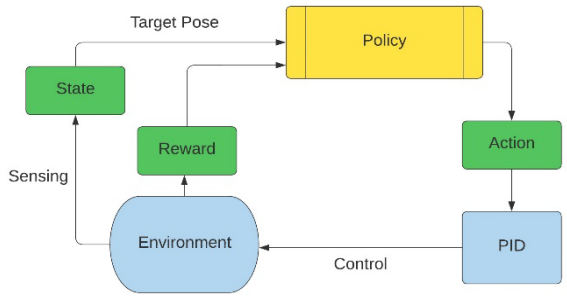


Figure 5. Overview of current architecture

To advance our research, we must first construct a simulation environment (such as a floor, a set of steps, or a flight of stairs), after which we must create the state and action spaces, the reward function, and other crucial components. Additional DRL-based algorithms are created and utilized to teach simulation-based policy. Finally, the taught policy is put into use on the virtual robot to carry out the given task.

Designing a new Approach

For the time being, the conventional approach TRPO is effective, although as was already mentioned, TRPO has significant drawbacks when dealing with complex problems. We just add a few additional algorithms to our conventional TRPO in order to address this. To effectively compare the estimated distribution to the normal distribution, we used KL divergence. We required a way to conveniently determine the sample space without going overboard on the computing cost due to a large batch size and higher number of episodes. The following is a formal definition of KL divergence,

$$D_{KL}(p||q) = \sum_{i=1}^{N} p(x_i)\log(\frac{p(x_i)}{q(x_i)})$$

Equation 3. Mathematical representation of KL divergence

The closer our approximation to the genuine distribution, the lower the KL divergence value. The vanishing gradient problem is a new difficulty brought on by traditional TRPO and KL divergence. We used a broad strategy to overcome this problem by employing a rectified linear unit (ReLU) activation function. The rectified linear activation function is a piecewise linear function that outputs zero otherwise and the input directly if the input is negative. Because a model that utilizes it is simpler to train and frequently performs better, it has evolved into the standard activation function for many different kinds of neural networks

Three components make up our final architecture: a gait planner, a gait controller, and a virtual PID. The state space, which describes the robot's state, and the action space, which is concerned with goal poise, make up the bulk of the gait planner architecture. A gait pattern is generated based on these variables. The Agent begins in a starting condition and then does an action in response to the observations. The transition function is used to calculate the observations by considering how actions affect the environment.

The reward function receives a feed from the transition function once it has interpreted the new state. Reward function inference, a decision requester module, and an action inference module make up the gait controller. The basic function of the reward function module is the pessimistic bound reward. Then it is modified by the reward function to maximize the reward from the acts.

The agent is then given this data once more, understands it, and decides what to do next. A domain randomization module and an adaptation module for the development of support for various robot configurations were later additions to the design.

Finalizing the Approach

A feedforward neural network with two input layers, twenty hidden layers, and one output layer makes up the final full model. 435,024 weights are embedded in it. Action masks and output layers produce the action, while input layers supply the vector observations. The 20 hidden layers are made up of 5 activation layers, 4 dense layers, 2 of each min, max, mul, and add layers, as well as 1 layer each of RandomNormal, Div, and Nop.

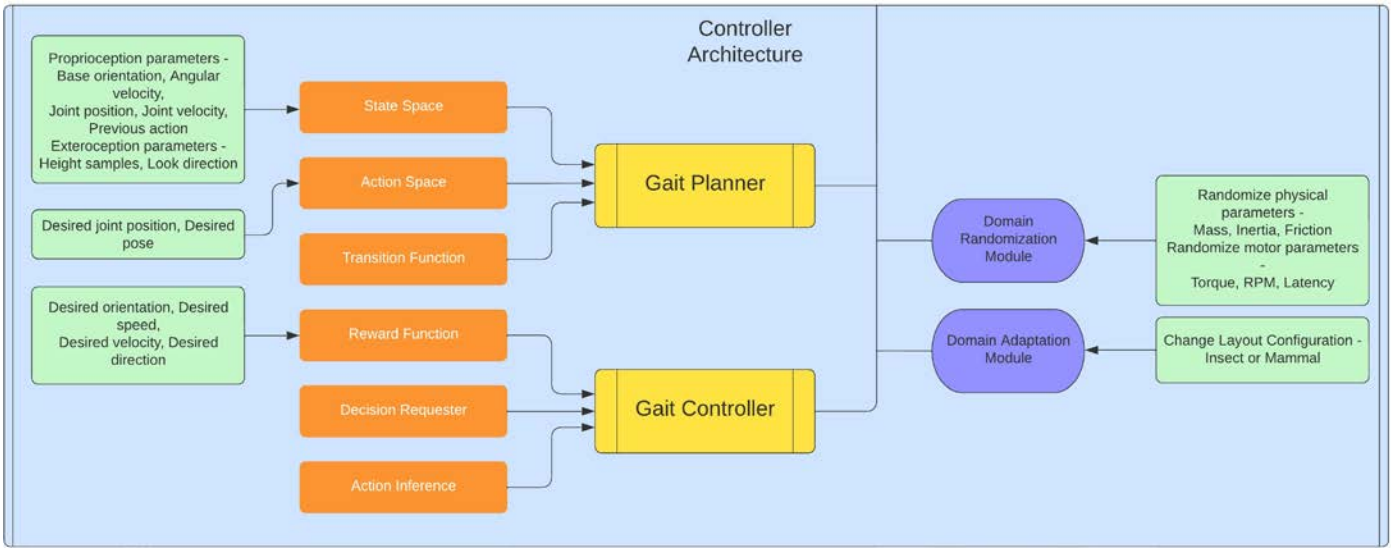


Figure 6. Controller architecture

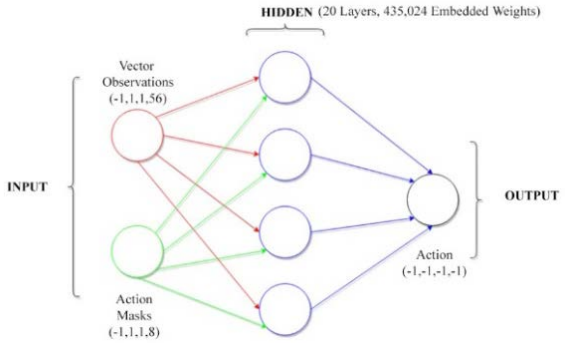


Figure 7. Generated neural model layout

We'll now move on to the analysis of our experimental findings for the system developed through the robot simulation in Unity Engine.

RESULTS AND DISCUSSION

In this section, we simulate our robot using the learned model as the controller to verify our methodology. To replicate the dynamic physical laws of the robot itself more realistically and effectively resolve the collisions caused when the robot interacts with the environment, we need a tool or program. The majority of academics have opted for the Pybullet and RaiSim simulation systems over the last few years. The precision of the robotic simulators used in academics today, however, is far lower than that of simulators used in video games, and they are still quite basic.

Common robotic simulators like Pybullet and RaiSim, which are extended for real-world simulations, can only address control-level simulations. They were created to function on CPUs with little parallel processing.

However, despite being a well-known DRL algorithm verification simulator, mujoco is rarely used as a platform for the deployment and testing of quadrupedal locomotion algorithms. As a result, we used the Unity game engine, a well-known but still relevant piece of software with a strong physics engine.

As for the actual training, we used a manual genetic technique. Agents were cloned in batches, trained in each batch, and the most productive batch was selected as the brain for the subsequent training phase. There were a total of three training phases, after which the outcomes were assessed.

Model Evaluation

Figure 8. Visualisation of Speed of Quadruped robot in Unity simulation

Figure 9. Visualisation of Height samples of Quadruped robot in Unity simulation

We build Unity-based graphs and plot our numbers in real-time to evaluate and track the performance of our model. We make a speed plot to help us track the discrepancy between the target speed and the robot's average measured speed, a height plot to track target and measured height samples, and an inclination map, as shown in Figure 8, 9 and 10.

Figure 10. Visualisation of the Quadruped robot's inclination in Unity simulation

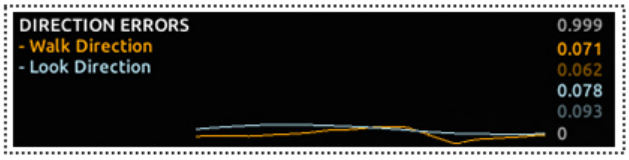


Figure 11. Visualisation of the Direction errors in Unity simulation

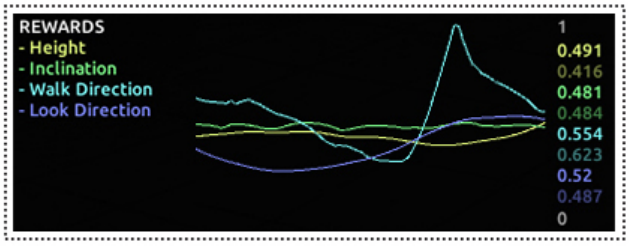


Figure 12. Visualisation of the Agent reward parameters in Unity simulation

We draw a graph that tracks the reward variables, net reward, and penalties for speed and directional faults in order to evaluate the system.

Figure 13. Visualisation of the reward penalties in Unity simulation

Figure 14. Visualisation of the Agent's reward sum in Unity simulation

The reward sum is then monitored in a final plot, as shown in Figure 14. Following are the different plots obtained via the tensorboard while training our agents. We used visual interpretation for evaluation as it will be much easier to evaluate the precision of agent and verify them.

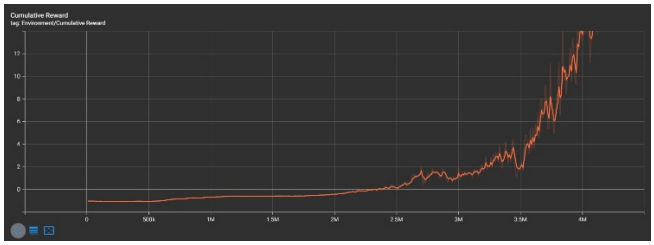


Figure 15. Visualisation of Cumulative reward via tensorboard

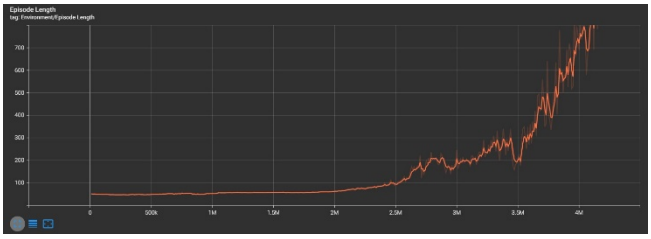


Figure 16. Visualisation of Episode length via tensorboard

Figure 15 represents the nature of cumulative reward with respect to step. As it can be clearly perceived that the cumulative reward for the agent increases with the increasing steps.

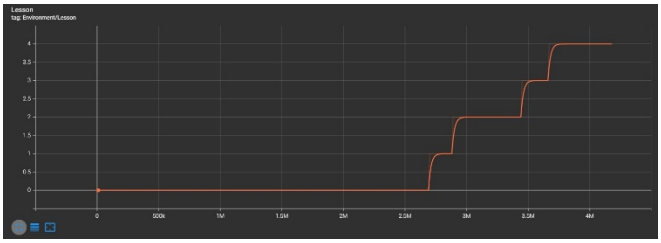


Figure 17. Visualisation of Lesson via tensorboard

Figure 17 represents the nature of lesson with respect to step. The lesson cycle for the agent increases with the increasing steps but in an odd interval bound manner. The first rise can be observed after the step 3.69 M and then can be observed to be increasing at irregular intervals.

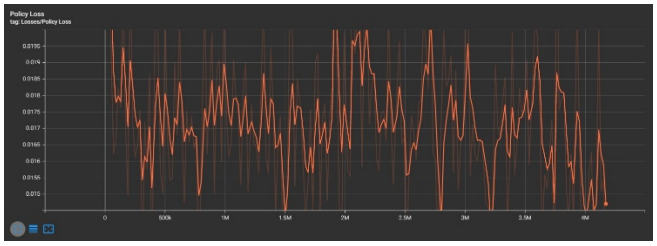


Figure 18. Visualisation of Policy loss in agent via tensorboard

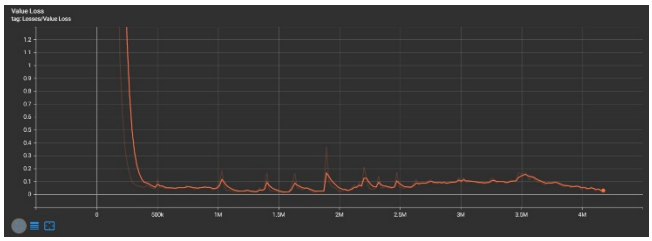


Figure 19. Visualisation of Value loss in agent via tensorboard

Figure 18 indicates the policy loss of the training agent in relation to the training steps. It can be observed to be decreasing with irregular spikes in between. Figure 19 indicates the value loss during the training session and can be clearly perceived as exponentially decreasing with minor spikes at irregular intervals.

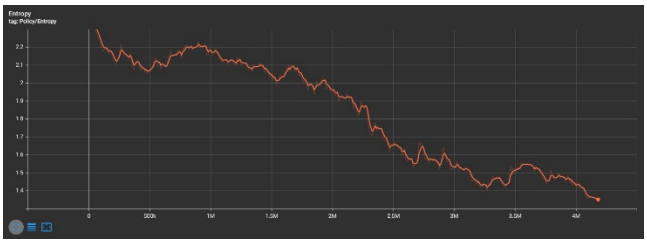


Figure 20. Visualisation of Entropy via tensorboard

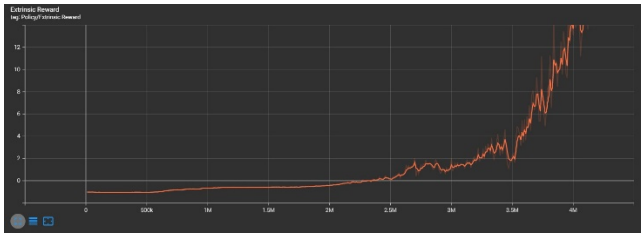


Figure 21. Visualisation of Extrinsic reward via tensorboard

Figure 20, 21, 22 illustrates the various characteristic features of policy with relation to the training steps. Figure 20 illustrates the entropy vs step relation and thus the inversely proportional nature between the two can be observed. The relation between extrinsic reward and step is varied as it follows an irregular incremental curve with an irregular decay rate. The same can be said for the estimate value as it follows the similar characteristic to its reward counterpart. It can be observed that there is an irregular growth pattern in this particular relation and thus cannot be generalized.

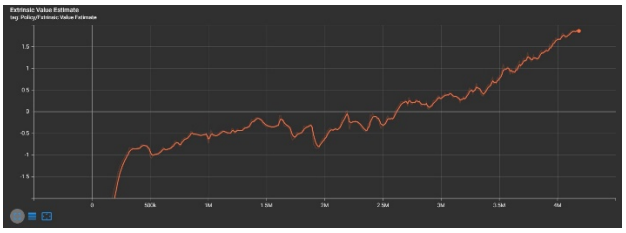


Figure 22. Visualisation of Extrinsic value estimate via tensorboard

The relationship between learning rate and step is a unique relation to ponder on, as it basically follows a linear decrement nature. Hence, it can be concluded that with the increasing number of steps the learning rate decreases as the smarter the agent becomes the less it requires to learn.

These all plots helped us to visualize the various relations between these crucial factors quite conveniently and thus also helped to monitor the status of our agent and its training performance.

Once the training is completed or a satisfactory epoch is achieved, the training is terminated and the model is saved as a .nn

file via the aid of tensorboard. This model is then used as the new brain for our agent and thus the agent can then control the gait efficiently.

This process is facilitated with the help of Barracuda extension for Unity and therefore simplifies the complex process for importing a neural network in the engine.

Overall, we find that the model uses the virtual robot to successfully pass the initial walking test during real-time execution. The significant difference between the actual speed and the target speed is a serious flaw. This can be enhanced during the reinforcement learning process by reducing the modelling error and adding more significant environmental disturbances. Better sensors, algorithms for analyzing sensor data, and larger networks are examples of additional techniques for improvement.

CONCLUSION

The rationale, design patterns, thorough implementation, and testing procedures for developing a DRL-based agent that can regulate the gait patterns of a quadruped robot are all described in this work. We may conclude that the model has done well and has shown to be relatively efficient for the task after having finished the analyses described in this paper. This project debuted a brand-new DRL-based quadruped robot gait controller system. It is introduced to provide the environment for mimicking robot movement. The entire system is broken down into a number of smaller systems, including the controller, PID, and gait planning modules. The environment was created using Unity Engine and has undergone experimental validation. It performs better than traditional control techniques in terms of precision and response time. Because it does not experience accuracy dilution, it can increase overall performance by a factor of up to 2.5 in our studied scenario.

This experiment demonstrated to us how AI-powered self-automation control systems may be built utilizing reinforcement learning. Since the agent is easily adaptable and can pick up new maneuvers in unfamiliar contexts, it may be trained for a straightforward situation like this one and then challenged to work in other situations. In the end, this research provides a solid foundation for further study into controls and human-machine interaction. Because it is modularly constructed, the system is extremely adaptable for future expansions and may be improved to make it more efficient.

Although the system is not yet ready for usage in any kind of production context, it still proves an important point. This method of developing an autonomous control system might be used in more practical applications to handle not only the motion of the robot in an open area but also other control inputs for various more intricate patterns. In conclusion, neural networks can be extended to control sophisticated robots. These could eventually be enhanced to carry out more intricate control schemes on bigger and more advanced robotic systems.

DECLARATION OF CONFLICTING INTERESTS

The author(s) declared no potential conflicts of interest with respect to the research, authorship, and/or publication of this article.

REFERENCES AND NOTES

1. Y. Zhong, R. Wang, H. Feng, Y. Chen. Analysis and research of quadruped robot's legs: A comprehensive review. *Int. J. Adv. Robot. Syst.* **2019**, 16 (3), 172988141984414.
2. M. Hutter, C. Gehring, D. Jud, et al. ANYmal - a highly mobile and dynamic quadrupedal robot. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*; IEEE, **2016**; Vol. 2016, pp 38–44.
3. Y. Duan, X. Chen, R. Houthooft, J. Schulman, P. Abbeel. Benchmarking deep reinforcement learning for continuous control. In *33rd International Conference on Machine Learning, ICML 2016*; **2016**; Vol. 3, pp 2001–2014.
4. M.M. Gor, P.M. Pathak, A.K. Samantaray, J.M. Yang, S.W. Kwak. Control oriented model-based simulation and experimental studies on a compliant legged quadruped robot. *Rob. Auton. Syst.* **2015**, 72, 217–234.
5. V. Tsounis, M. Alge, J. Lee, F. Farshidian, M. Hutter. DeepGait: Planning and Control of Quadrupedal Gaits Using Deep Reinforcement Learning. *IEEE Robot. Autom. Lett.* **2020**, 5 (2), 3699–3706.
6. K. Xu, H. Ma, J. Chen, et al. Design and Analysis of a Metamorphic Quadruped Robot. In *2018 International Conference on Reconfigurable Mechanisms and Robots, ReMAR 2018 - Proceedings*; **2018**.
7. L. Yao, H. Yu, Z. Lu. Design and driving model for the quadruped robot: An elucidating draft. *Adv. Mech. Eng.* **2021**, 13 (4).
8. G. Kenneally, A. De, D.E. Koditschek. Design Principles for a Family of Direct-Drive Legged Robots. *IEEE Robot. Autom. Lett.* **2016**, 1 (2), 900–907.
9. S. Kitano, S. Hirose, G. Endo, E.F. Fukushima. Development of lightweight sprawling-type quadruped robot TITAN-XIII and its dynamic walking. In *IEEE International Conference on Intelligent Robots and Systems*; **2013**; pp 6025–6030.
10. P. Biswal, P.K. Mohanty. Development of quadruped walking robots: A review. *Ain Shams Eng. J.* **2021**, 12 (2), 2017–2031.
11. J. Zhang, J. Zhang, C. Wang. Dynamic analysis and simulation on bionics quadruped robot. *Open Autom. Control Syst. J.* **2015**, 7 (1), 1088–1093.
12. X. Sang, G. Du, H. Wu, D. Zhang. Gait Analysis and Realization of Quadruped Bionic Robot with 8 Degrees of Freedom. In *2021 IEEE International Conference on Advances in Electrical Engineering and Computer Applications, AEECA 2021*; **2021**; pp 90–93.
13. S. Zhang, M. Fan, Y. Bin Li, X. Rong, M. Liu. Generation of a continuous free gait for quadruped robot over rough terrains. *Adv. Robot.* **2019**, 33 (2), 74–89.
14. L. Ding. Key technology analysis of BigDog quadruped robot. *Jixie Gongcheng Xuebao/Journal Mech. Eng.* **2015**, 51 (7), 1–23.
15. M.H. Rahman, M.M. Islam, M.F. Al Monir, et al. Kinematics analysis of a quadruped robot: Simulation and Evaluation. In *2022 2nd International Conference on Image Processing and Robotics, ICIPRob 2022*; **2022**.
16. J. Hwangbo, J. Lee, A. Dosovitskiy, et al. Learning agile and dynamic motor skills for legged robots. *Sci. Robot.* **2019**, 4 (26).
17. J. Che, Y. Pan, W. Yan, J. Yu. Leg Configuration Analysis and Prototype Design of Biped Robot Based on Spring Mass Model. *Actuators* **2022**, 11 (3).
18. C. Gonzalez, V. Barasuol, M. Frigerio, et al. Line walking and balancing for legged robots with point feet. In *IEEE International Conference on Intelligent Robots and Systems*; **2020**; pp 3649–3656.
19. K. Michael. Meet Boston Dynamics' LS3 - The Latest Robotic War Machine. *Fac. Eng. Inf. Sci. - Pap.* **2012**, Part A, 2773.
20. C. Yang, K. Yuan, Q. Zhu, W. Yu, Z. Li. Multi-expert learning of adaptive legged locomotion. *Sci. Robot.* **2020**, 5 (49).
21. K. Mitobe, N. Mori, K. Aida, Y. Nasu. Nonlinear feedback control of a biped walking robot. In *Proceedings - IEEE International Conference on Robotics and Automation*; **1995**; Vol. 3, pp 2865–2870.
22. K. Xu, S. Wang, B. Yue, et al. Obstacle-negotiation performance on challenging terrain for a parallel leg-wheeled robot. *J. Mech. Sci. Technol.* **2020**, 34 (1), 377–386.
23. S. Ma, T. Tomiyama, H. Wada. Omnidirectional static walking of a quadruped robot. *IEEE Trans. Robot.* **2005**, 21 (2), 152–161.

24. E. Garcia, J.C. Arevalo, G. Mũnoz, P. Gonzalez-de-Santos. On the biomimetic design of agile-robot legs. *Sensors* **2011**, 11 (12), 11305–11334.
25. L. Wang, C. Wang, W. Du, et al. Parameter optimization of a four-legged robot to improve motion trajectory accuracy using signal-to-noise ratio theory. *Robot. Comput. Integr. Manuf.* **2018**, 51, 85–96.
26. J. Guzzi, R.O. Chavez-Garcia, M. Nava, L.M. Gambardella, A. Giusti. Path Planning with Local Motion Estimations. *IEEE Robot. Autom. Lett.* **2020**, 5 (2), 2586–2593.
27. P.M. Wensing, A. Wang, S. Seok, et al. Proprioceptive actuator design in the MIT cheetah: Impact mitigation and high-bandwidth physical interaction for dynamic legged robots. *IEEE Trans. Robot.* **2017**, 33 (3), 509–522.
28. L. Liu, X. Liu, C. Zhang. Realization of Neural Network for Gait Characterization of Quadruped Locomotion∗. *J. Appl. Anal. Comput.* **2022**, 12 (2), 455–463.
29. C. Lee, D. An. Reinforcement learning and neural network-based artificial intelligence control algorithm for self-balancing quadruped robot. *J. Mech. Sci. Technol.* **2021**, 35 (1), 307–322.
30. R.S. Sutton, A.G. Barto. Reinforcement Learning: An Introduction; MIT Press, **1998**.
31. S. Tzafestas, M. Raibert, C. Tzafestas. Robust sliding-mode control applied to a 5-link biped robot. *J. Intell. Robot. Syst. Theory Appl.* **1996**, 15 (1), 67–133.
32. Y. Jia, X. Luo, B. Han, et al. Stability criterion for dynamic gaits of quadruped robot. *Appl. Sci.* **2018**, 8 (12).
33. Q. Hao, Z. Wang, J. Wang, G. Chen. Stability-guaranteed and high terrain adaptability static gait for quadruped robots. *Sensors (Switzerland)* **2020**, 20 (17), 1–26.
34. J.P. Chen, H.J. San, X. Wu, B.Z. Xiong. Structural design and gait research of a new bionic quadruped robot. *Proc. Inst. Mech. Eng. Part B J. Eng. Manuf.* **2022**, 236 (14), 1912–1922.
35. M. RS. Test and evaluation of a versatile walking truck. In *Off-Road Mobility Research Symposium, International Socity for Terrain Vehicle Systems*; **1968**; pp 359–379.
36. J. Estremera, K.J. Waldron. Thrust control, stabilization and energetics of a quadruped running robot. *Int. J. Rob. Res.* **2008**, 27 (10), 1135–1151.