MODEL PREDICTIVE CONTROL BASED JUMPINGOF ROBOTIC LEG ON A PARTICULAR HEIGHT USING REINFORCEMENT LEARNING

Amrita Rai[-1], Shyam Dwivedi[-2]

[1,2]*Department of Computer Science and Engineering, Rameshwaram Institute of Technology andManagement, Dr A PJ Abdul Kalam Technical University, Lucknow, India*

Abstract — A robotic limb that is controlled by an MPC and trained via reinforcement learning in a real-worldsetting will attempt to learn how to jump beyond a certain height. A dynamic controller, model predictive control regulates each process by resolving constraints. Moreover, reinforcement learning will assist in developing the robotic leg's capacity to adapt in situations that change rapidly.

For regulating the dynamic jumping of a robotic limb through the control design, the authors of this study provide the reinforcement learning with model predictive control approach. In this situation, MPC are primarily utilized to regulate the torque, angles, positions, frictions, force, and other dynamic characteristics on joints. According to this work, hopping might be taught to a robotic leg using model predictive reinforcement learning in a practical environment. Model-based reinforcement learning includes a component called model, which refers to predetermined Kinematics and Dynamics parameters. PPO (proximal policy optimizer) and A2C (advanced actor-critic) algorithms are used in reinforcement learning. Recent years have seen the discovery of predefined forward and inverse kinematics and dynamics parameters that can assist Robotic leg to precisely jump over certain height. The first section is the introduction, which includes a discussion of recent study findings as well as a review of all relevant literature. A brief explanation of each phrase is covered in the second section, while the third section discusses its applications, workings, and formula. The fourth portion is the conclusion, and the final part is a list of sources used in this study.

1. INTRODUCTION

1.1 Robots

A Robots are machines that can be programmed by computers and are capable of per- forming a complex series of tasks on their own. An external control device or an internal control system can both control a robot. Even though some of them are made to resemble people, most robots are task-performing machines that put basic usefulness before emotive aesthetics. The design, maintenance, implementation, and use of robots as well as the computer systems that control, sense, and process data for them are all covered by the field of technology knownas robotics. These technologies deal with automated systems that can replace people in hazardous situations or during production processes or that are similar to people in terms of behaviour, cognition, or appearance.

Robots' ability to perform simple tasks in a manner similar to that of humans makes their work simple, quick, and easy. There are far too many domestic robots in our daily lives. People are rapidly growing more dependent on robots because they do household tasks. Some of the most well-liked robots are vacuum cleaners and kitchen robots, but we also have robots that can mow the lawn, clean the windows, and iron your clothes. Robots will someday take over certain human tasks due to their ever-improving skills, but not all. Robotics can presently only automate 25 percent of occupations in unstable, human-dependent industries like nursing and construction. Robots, however, are and always will be dependent on human programming.

Robots are automated machines that require little or no human assistance to complete specified jobs quickly and accurately. Over the past 50 years, there have been significant advancements achieved in the field of robotics, which is focused on the design, creation, and application of robots.

Essentially, there are as many different types of robots as there are tasks for them to perform. While some tasks are better performed by people rather than machines, others are better left to them.

Robots are more adept at the following tasks than humans are:

1. In business or industrial settings, automate manual or repetitive processes.

2. Perform dangerous or unpredictable tasks in order to find dangers like gas leaks.
3. Deliver and process reports on corporate security.
4. Fill out prescriptions for medication and prepare IVs.
5. Deliver internet orders, room service, and even food packages in an emergency.
6. Help patients throughout operations.

Figure 1.1: Boston Robots

Currently, industrial robots are utilised to transport massive boxes and items in many different fields. Robotic hands and legs are only two examples of the many robots utilised in the medical field. This is great news for persons with disabilities. Robots are essential to the study of physics because they can perform tasks that humansare unable to. We are specifically referring to space robots, which have the ability to live in space, transport supplies for space missions, monitor space stations, or just stroll on the earth so that people may view these locations from a distance. Industrial robots are the most well-known and practical robots. An industrial robot is a versatile manipulator that can be autonomously controlled, reprogrammed, and programmed in three or more axes.These robots can also be customised by users for various uses. Businesses have been able to perform higher-leveland more sophisticated jobs by combining these robots with AI, going beyond mere automation.

1.2 Legged robot

Legged robots are a subset of mobile robots that move by using movable limbs, such as leg mechanics. They can move across a wider range of surfaces than wheeled robots and are more adjustable, but these advantages come at the expense of more difficulty and power consumption. As an example of bio-mimicry, legged robots frequently mimic other legged species, including humans or insects.

One-legged, two-legged, four-legged, six-legged, eight-legged, and hybrid robots are just a few of the numerous varieties of legs that exist today.

They are walking robots that move by controlling their legs. They are employed to provide movement in extremely unstructured settings. Despite being difficult to construct, they have an advantage over wheeled robotswhen it comes to navigation on any kind of surface or path. Robots with legs are more adaptable than those with wheels; they can move across a variety of surfaces. As an illustration of bio-mimicry, they can mimic animals with legs, such as people or insects. Robots with legs are capable of navigating on any surface.

The gaits that are possible for them can be classified according to how many limbs they employ. Robots with more legs may be more stable, whereas those with fewer legs are more manoeuvrable. Robots with four legs are commonly known as quadruped robots or “Big Dogs.” They are designed to navigate challenging terrain. They are capable of moving on four legs. Compared to two-legged robots, they can benefit from greater stability, especially when moving. Compared to two-legged systems, they can benefit from having a lower centre of gravity.

One of Boston University’s top quadruped robots, the Mini Cheetah, has four legs. It is a dynamic quadrupedal robot that can move quickly over a variety of surfaces. It is the ideal fusion of electrical, mechanical, and computer science.

Figure 1.2: mini cheetah

The Mini Cheetah can move in a variety of gaits over various types of hard terrain, and to maintain balance it needs to manage its leg actuators, sensors to detect where its feet should be, and planning algorithms to choose its course and speed.

1.3 Reinforcement Learning

A branch of machine learning called reinforcement learning (RL) studies how intelligent creatures should behave in a given environment to maximise the concept of gaining prizes. In reinforcement learning, an agent picks up information from their interactions with the environment in order to amass a cumulative reward without being watched. By taking actions and monitoring the results of those actions, an agent learns how to function in a given environment using the feedback-based machine learning process known as reinforcement learning. Each successful action results in good feedback for the agent, and each unsuccessful action results in negative feedbackor a punishment. Using RL, it is possible to tackle a specific class of issues, including those in robotics and gamingwhere choices must be made sequentially and with a long-term goal in mind.

Computers can be taught to learn using the reinforcement learning method, which involves rewarding or punishingdesired behaviour. A reinforcement learning agent can monitor and comprehend its environment, act, and learn from errors. The agent learns new information by trial-and-error learning, and as it gains experience, it improvesits performance abilities. Reinforcement learning is a type of machine learning technique because it involves an intelligent agent (computer programme) interacting with the environment and learning how to function within it. The way a robotic dog learns to move his arms is a good illustration of reinforcement learning.

The environment is typically described as a Markov decision process since many reinforcement learningalgorithms for this case use dynamic programming techniques (MDP).

The main difference between traditional dynamic programming techniques and reinforcement learning algorithmsis that the latter are more suited to large MDPs than exact procedures are and do not necessitate knowledge of a precise mathematical representation of the MDP.

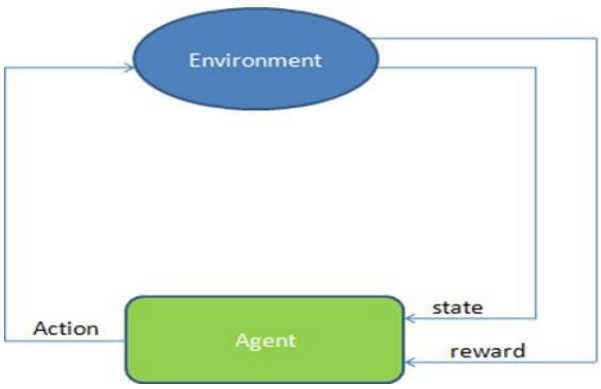

Figure 1.3: reinforcement Learning

The illustration that follows shows reinforcement learning in action. An agent behaves in the environment to achieve a purpose. The agent either works +1 or -1 depending on the action reward function. An action is performed on a state. Once the agent obtains a reward proportionate with their acts, the next state will be held, and so on until the agent achieves their goal.

Reinforcement learning algorithms look for a policy based on previous experiences to direct an agent towards achieving a specified objective. To model this task, the Markov Decision Process (MDP) is employed. The variables S, A, P, and r make up the tuple that represents the MDP. S Rn stands for the state space, A Rm for theset of actions the agent is capable of, P: S A 7 S for the system dynamics, [0, 1] for the discount factor, and r: S A 7 R for the reward function that either rewards or penalises an action performed in state.

1.4 Theoretical Background

1.4.1 Dynamics

The two types of dynamics are linear dynamics and rotational dynamics. Linear dynamics takes into account variables like force, mass/inertia displacement (measured in distance units), velocity (proxied in path length per unit of time), acceleration (measured in distance per unit of time squared), and momentum when it comes to moving objects in a straight line (mass times unit of velocity). Torque, moment of inertia/rotational inertia, angulardisplacement (in radians or, less frequently, degrees), angular velocity (in radians per unit time), angular acceleration (in radians per unit of time squared), and angular momentum are certain aspects of rotational dynamics (moment of inertia times unit of angular velocity). Objects frequently travel straight and in both directions. Moving from one place to another, there are velocity, position, acceleration, torque angular velocity, inertia are working together. Dynamics are really play a vital role to move any body and we are working on a legso here dynamics are so much important.

1.4.2 Kinematics

Astrophysics uses kinematics to explain how individual celestial entities and groups of them move. An engine, a robotic arm, or the human skeleton are examples of multi-link systems, and the motion of these systems is referred to as kinematics. It is utilised in biomechanics, robotics, and mechanical engineering. To describe the spatial positions of bodies or systems of material particles, as well as their speeds and velocities, is the aim of kinematics (acceleration). When the generating factors are disregarded, motion descriptions are only plausible for particles with restricted motion or particles moving along routes. In uncontrolled or free motion, forces govern the shape of the path.

1.4.2.1 Forward Kinematics

Forward kinematics is the process of using a robot's kinematic equations to ascertain the joint parameters that must match the end-position effector's prescribed values. From the base frame to the end effector, a manipulatoris constructed of serial links that join to one another at revolute or prismatic joints. Forward kinematics describes the location and orientation of the end effector in relation to the joint variables. A proper kinematics model shouldbe used in order to have forward kinematics for a robot mechanism in a systematic method. The D-H parameter, which contains four parameters, is utilised. The values ai1, alpha, 1idi, and theta, respectively, describe the link length, twist, offset, and joint angle. A connected coordinate frame is used.

To determine DH parameters for each joint. There is a method for computing forward kinematics, and the coordinate frame's z axis points along the general manipulator's rotational or sliding direction. D-H Parameter:

The four parameters known as Denav t-Harte berg parameters, or DH parameters, are connected to a specific convention for linking reference frames to the links of a robot manipulator or spatial kinematic chain.

The following four transformation parameters are known asD–H parameters:

d: offset along previous z to the common normal. D is the distance alongiJ0 Z axis to translate. Theta: angle about previous z, from old x to new x. Theta is the angle to rotate around the J0 Z axis.L: length of the common normal (link length).

Alpha: angle about common normal from old z axis to new z axis. Alpha is the angle to rotate around the J0 X axis.

1.4.2.2 Inverse Kinematics

The end of a kinematic chain, such as the skeleton of an animated character or a robot manipulator, needs to be placed in a specific position and orientation with respect to the start of the chain. Inverse kinematics is a mathematical technique that establishes the variable joint parameters required for this.

By employing kinematic equations, inverse kinematics can be used to predict a robot's path to a given point. As an illustration, inverse kinematics is used to determine the necessary location for a leg to jump or slip. Serial manipulators' inverse kinematics problem has long been researched. Having control over manipulators is crucial.In the real-time control of manipulators, solving the inverse kinematics frequently requires a significant amount of processing and takes a very lengthy time. Actuators work in joint space, whereas manipulators complete their jobs in Cartesian space. The orientation matrix and the location vector are both parts of Cartesian space. Contrarily, joint angles are a representation of joint space. Inverse kinematics refers to the difficulty of translatingthe location and orientation of a manipulator end effector from Cartesian space to joint space. To determine the analytical inverse kinematics solution, two techniques are used: geometric and algebraic.

1.4.3 Deep reinforcement learning

A branch of machine learning called deep reinforcement learning (deep RL) combines deep learning and reinforcement learning (RL). In RL, the issue of a computational agent learning to make judgements through trialand error is taken into account. Because deep learning is incorporated into the solution, agents can make judgements based on input from unstructured data without having to manually engineer the state space. Deep RLalgorithms can decide what actions to take to achieve an objective by analysing very massive inputs, such as everypixel presented to the screen in a video game (maximising the game score). Robotics, video games, natural language processing, computer vision education, transportation, finance, and healthcare are just a few of the diverse fields in which deep reinforcement learning has been applied.

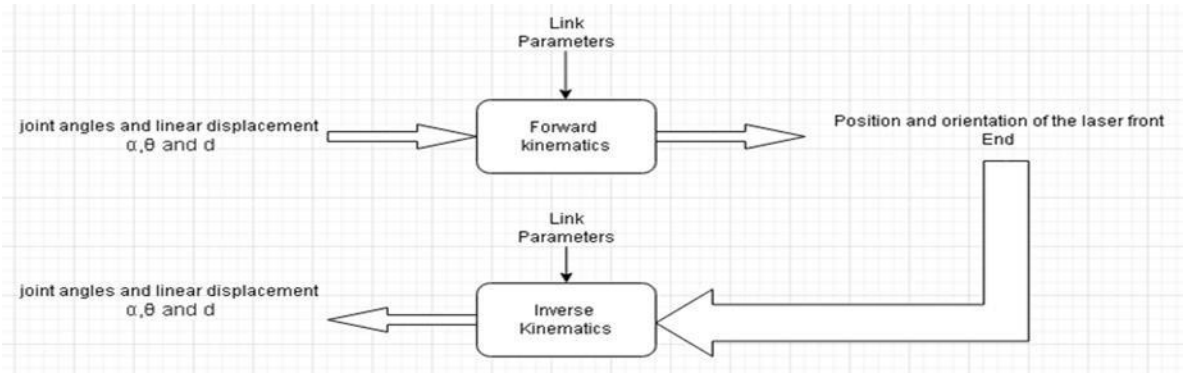

Figure 1.4: Block diagram of forward and inverse kinematics

2. Literature Review

2.3.1 State of the art

Diego Rodriguez and Suen Banked offer a unique Deep Reinforcement method in January 2021 that allows an agent to learn omnidirectional locomotion for humanoids and accomplishes the locomotion behaviour with a single control policy (a single neural network). the ability of the learnt policy to turn around the vertical axis at various speeds and walk in the sagittal and lateral orientations. A system utilising deep reinforcement learning isproposed by Guillanme Bellegarda and Quan Nguyen in March 2021. to benefit from the challenging quadrupedal jumping nonlinear trajectory optimization solution. While the "low" gain baseline performance is inferior to the "high" gain controller under ideal circumstances because of the large rise in noise. Julian Ibaz, Jie Tan, and Chelsea Finn hypothesised in February 2021 that robots can learn in a real-world context and Physical robots may now learn complicated skills in the actual world thanks to deep reinforcement learning, which has also shown promise in this regard. All forms of movement, employing reinforcement learning to learn behaviour in a real environment.

Using discriminatively trained exemplar models, Justin Fu, John Dco-Reyes, and Sergey proposed a novelty detection algorithm for exploration in May 2017. The classifiers are trained to distinguish between each visited state and all others. a scalable exploration method based on developing discriminative exemplar models to award novelty bonuses and reveal a novel relationship between exemplar models and density estimates. An automated learning environment for generating control rules directly on the hardware of a modular legged robot is presented by Sehon Ha, Joohyung Kim, and Katsu Yamane in august 2018. By computing the rewards using a vision-based tracking system and repositioning the robot to the starting place using a resetting mechanism, this environment supports the reinforcement learning processes.

A sample effective Deep reinforcement learning algorithm based on mechanism entropy reinforcement learning was proposed in June 2019 by Tumas Haarnoja, Sehoon Ha, Aurick Zhou, Jie Tan, George Tucker, and Sergey Levine. It only needs a small number of trials to train neural network policies and little per-task tuning. a whole end- to-end learning system for legged robot locomotion. Ted Xiao, Eric Jang, Dmitry Calash kav, Sergey Levine,Julian Ibarz, Karol Hausman, and Alexander Herzog will be in April 2020. According to this statement, a robot must think and move simultaneously, choosing its next course of action before the previous one has finished, much like a person or an animal. This is accomplished in reinforcement learning settings where sampling an action from the policy must be done concurrently with the time evolution of the controlled system.

Quan Nguyen, Mathew J. Powell, Benjamin Katz, Jared Di Carlo, and Sangbae Kim proposed a novel method in August 2019 for implementing optimum jumping behaviours on quadruped robots. This technology incorporates strong landing controllers for sustaining the robot body position and orientation after impact, accurate high- frequency tracking controllers, and effective trajectory optimisation. In addition to proving the advantages of the technology, the experimental findings demonstrate that the hardware can handle demanding tasks requiring high power production and high impact absorption. Zhaoming Xie and Patrick Clary describe and document an iterative design technique in March 2019 that takes into account the numerous reward design iterations that are frequently required in practise.

Transfer learning is accomplished throughout the processes by using deterministic action stochastic state tuples, which represent the deterministic policy action linked to the states visited by the stochastic policy. An iterative design process that uses RL to learn robust locomotion policies, the authors show that learned policies can be robust without resorting to dynamics randomization. A novelty detection system for exploration based on discriminatively trained exemplar models is described by Fu, John, and Sergey in 2017. a scalable exploration method based on training discriminative exemplar models that shows a novel relationship between exemplar models and density estimates while also awarding novelty bonuses. In order to provide novelty bonuses and showa novel relationship between exemplar models and density estimation, a scalable exploration technique based on training discriminative exemplar models is used.[15] In 2020, Xiao, Kalashnikov, Levin, and colleagues used reinforcement learning to choose their next course of action before the previous one had finished, and they discovered that concurrent continuous-time and discrete-time Bellman operators continued to contract, maintaining the Q-learning convergence guarantees. With the aid of a quadruped robot, you may learn policies for a wide range of behaviours that can then be effectively applied in the actual world. Bellman operators that were contemporaneous in continuous and discrete time remained contractions, maintaining the guarantees of Q- learning convergence.

Legged robots can acquire agile movement techniques by imitating real-world animals according to an imitation learning system that Peng and colleagues introduce in 2020. They have discovered a way to efficiently translate policies learned in simulation to the actual world by teaching a quadruped robot how to do a range of actions. Standard off-policy deep reinforcement learning algorithms like DQN and DDPG, as demonstrated by Fujimoto and Meger et al. in 2019, are useless for this fixed batch situation.

3. RESEARCH METHODOLOGY

A. Model Predictive Control

There are numerous dynamics and kinematics that will function on the robotic leg to jump over a specific height. To accomplish the purpose, we can formulate specific dynamics, such as torque, velocity, position, and angles, as well as kinematics, such as forward kinematics for position and inverse kinematics for joint angles. Leg will follow the state trajectory in order to jump over a specific height, and MPC will iteratively adjust the inputs as necessary to cause the leg to jump over the specified height. MPC is a controller that foresees how a robotic limb will behave in the future. In Order to determine the best control action for jumping at a specific height, it solves constraints based on robotic legs and optimization algorithms.

B. Reinforcement learning

Reinforcement learning will be used on that robotic leg to teach it how to jump at a given height in realistic settings. A reward system will function in accordance with the feedback. PPO (proximal policy control) and A2C (advanced actor critic) algorithms are utilized for training that leg. By taking part in an action and receiving feedback, an actor learns in a real-world setting called reinforcement learning. Robotic legs that jump or are readyto jump will be rewarded, otherwise they will receive no reward or a negative reward. Leg's environment will alter dynamically, and reinforcement learning will assist in helping them adopt the terms and circumstances whileattempting to reach the desired height.

4. REINFORCEMENT LEARNING ALGORITHM

The agent gains knowledge through trial and error, and based on its experience, it acquires the abilities required to complete the mission more successfully. In light of this, we may say that "Reinforcement learning is a type of machine learning method where an intelligent agent (computer programme) interacts with the environment and learns to function within that," The way a robotic dog learns to move his arms is a good illustration of reinforcement learning. These reinforcement learning techniques can assist the agent in learning and generating its best output.

I. PPO

A policy gradient method for reinforcement learning is called proximal policy gradient. It is a recent development in reinforcement learning that offers improvements to the optimization of Trust region policies (TRPO). We shallfirst grasp what policy stands for in order to comprehend the algorithm. In terms of reinforcement learning, a policy is from the action space to the state space. A reinforcement learning agent's policy determines which courseof action to pursue based on the current condition of the environment. When we talk about evaluating the policy function of an agent, we mean determining how well the agent is performing. This is where the policy gradient method comes into play. when an agent is learning but is not aware of the optimum course of action for the associated states. Numerous actions are used as input, and the greatest action will be the outcome, much like a neural network. PPO is an algorithm that strikes a compromise between key elements such implementation and testing ease, sample complexity and efficiency, and step-function updates. In actuality, PPO is a policy gradient method that also learns from online data. The Actor-critic model, which employs two deep neural networks, Actorand Critic, is the most popular way to implement PPO.

Lclip (θ) = El [min (rt (θ) At, clip (rt (θ), 1 − ϵ, 1 + ϵ) At)]

Et = empirical expectation overtime −steps.rt = ratio of the probabilities under then ew and old policy. (estimate advantage at time t.ϵ = hyper − parameter, usually0.1or0.2.

For PPO, there are significant implications or facts. 1.A policy-compliant algorithm is PPO.

2. In contexts with discrete or continuous action spaces, PPO can be applied.

3. MPI parallelization is supported by the Spinning Up implementation of PPO.

4. It uses a policy gradient optimization algorithm, which updates an existing policy in each step to try toimprove certain parameters.

5. It makes sure that the update is not too significant, meaning that the old and new policies are not drastically different.

6. ϵ is a hyper – parameter denotes the limit of the range within which the update is allowed.

Input: initial policy parameters. for iteration = 1, 2, 0, 0, 0. do

for actor = 1, 2, 0, 0, 0 do

Run policy π θo L din environment f or T time –steps compute advantage estimate A1, , Aiend f or

Optimize surro gate L with respect to θ, with k epochs and mini – batch them <= NTθ0ld < − − − − −θ

end for

What we can see is that small batches of observations, sometimes known as "mini- batch," are used for updatingand then discarded in order to make room for a fresh batch of data. To prevent large revisions that could potentiallybe irreparably detrimental, the new policy will be "-clipped" to a tiny zone. In other words, PPO operates just likeother policy gradient methods in that it computes the gradients to enhance the judgements or probabilities in the backward pass and computes the output probabilities based on various parameters in the forward pass. Like its predecessor, TRPO, it makes use of importance sampling ratio. However, it also makes sure that excessively substantial modifications are not permitted and that the old policy and new policy are at least within a specific proximity (denoted by). One of the most popular policy optimization techniques in the reinforcement learning space is now this one.

2.A2C

Advantage The actor-critic algorithm is a value-based algorithm that learns how to choose predictive actions or input actions. It is a policy-based algorithm that covers the mapping of input state to output state. To solve a particular problem in the actor critic algorithm, the actor and critic networks work together. The agent (temporal difference) or prediction error is determined by the advantage function. Learning a hybrid approach (A2C) or a Reinforcement Learning algorithm that combines value optimization with policy optimization techniques. A performer of an action can learn which of their activities are better or worse by receiving feedback on them. Temporal difference learning agents learn by predicting rewards in the future and altering their strategy in reaction to faulty predictions. One of the remarkable things about temporal difference learning is that it appears to be oneof the ways the brain picks up new information.

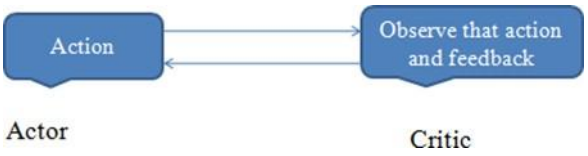

Figure: A2C Working

As is well known, a neural network can be used to parameterize the Q function and learn the Q value. This bringsup actor-critic techniques, where:

1.The value function is estimated by the "Critic." The action-value (the Q value) or state-value (the V value) might be this. 2. Critic: A Q-learning algorithm that evaluates the action that the Actor chose and offers suggestions forimprovement. It can benefit from Q-learning efficiency techniques like memory replay. At each update phase, weupdate both the Value network and the Critic network.

This indicates how much better it is to take a particular action than the typical general action at the current state. We therefore subtract the Q value part from the Value function while utilising it as the baseline function. This iswhat we'll refer to as the advantage value:

A (St, at) = Q (St, at) − v(at)Where,

at = action taken as per time St = State of an action

and here pseudo code for A2C algorithm for solving the problems based on reinforcement learning.

Algorithm 1 Asynchronous one-step Q-learning - pseudocode for each actor-learner thread.

// Assume global shared θ, θ^-, *and counter* $T = 0$.
Initialize thread step counter $t \leftarrow 0$
Initialize target network weights $\theta^- \leftarrow \theta$
Initialize network gradients $d\theta \leftarrow 0$
Get initial state s
repeat
  Take action a with ϵ-greedy policy based on $Q(s, a; \theta)$
  Receive new state s' and reward r
  $y = \begin{cases} r & \text{for terminal } s' \\ r + \gamma \max_{a'} Q(s', a'; \theta^-) & \text{for non-terminal } s' \end{cases}$
  Accumulate gradients wrt θ: $d\theta \leftarrow d\theta + \frac{\partial (y - Q(s,a;\theta))^2}{\partial \theta}$
  $s = s'$
  $T \leftarrow T + 1$ and $t \leftarrow t + 1$
  if $T \mod I_{target} == 0$ **then**
    Update the target network $\theta^- \leftarrow \theta$
  end if
  if $t \mod I_{AsyncUpdate} == 0$ or s is terminal **then**
    Perform asynchronous update of θ using $d\theta$.
    Clear gradients $d\theta \leftarrow 0$.
  end if
until $T > T_{max}$

Figure: Pseudocode for A2C

V. CONCLUSION

In this paper, we introduced a novel methodology for robotic leg to jump over a particular height using model predictive control with reinforcement learning. Model predictive control a method of formulation for every dynamic manoeuver at every joint of robotic leg. Reinforcement learning is an algorithm to train that leg in real environment for hoping over a particular height. After conducting experiments in order to accomplish the goal and analysed the findings to reach the following conclusions:

a) By employing reinforcement learning, a model-based strategy, and no control system, model will be train to jump beyond a specified height.

b) Reinforcement learning lead us to explore feasible implementation of highly dynamic movements without anyequation of motion.

c) Using reinforcement learning we can achieve higher height.

REFERENCES

[1] D. Rodriguez and S. Banked Deep walk omnidirectional bipedal gait by deep reinforcement learning, in2021 IEEE International Conference on Robotics and Automation (ICRA). IEEE, 2021, pp. 3033–3039.

[2] G. Bellegarda and Q. Nguyen, "Robust quadruped jumping via deep reinforcement learning," arc Xiv preprint arci: 2011.07089, 2020.

[3] Y. Ding, A. Mandala, C. Li, Y.-H. Shin, and H.-W. Park, "Representation- free model predictive control for dynamic motions in quadrupeds," 12 2020.

[4] J. Di Carlo, P. M. Wincing, B. Katz, G. Belt, and S. Kim, "Dynamic locomotion in the mit cheetah 3 through convex model-predictive con- troll," in 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2018, pp. 1–9.

[5] J. Ibarz, J. Tan, C. Finn, M. Kalakrishnan, P. Pastor, and S. Levine, "How to train your robot with deep reinforcement learning: lessons we have learned," The International Journal of Robotics Research, vol. 40, pp.698 – 721, 2021.

[6] T. Haarnoja, A. Zhou, S. Ha, J. Tan, G. Tucker, and S. Levine, "Learning to walk via deep reinforcementlearning," Ar Xiv, vol. abs/1812.11103, 2019.

[7] Z. Zhang, "Model predictive control of quadruped robot based on reinforcement learning," AppliedSciences, vol. 13, p. 154, 12 2022.

[8] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. M. O. Hess, T. Erez, Y. Tassa, D.Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," CoRR, vol. abs/1509.02971, 2016.

[9] S. Fujimoto, D. Meger, and D. Precup, "Off-policy deep reinforcement learning without exploration," inICML, 2019.

[10] T. Haarnoja, H. Tang, P. Abele, and S. Levine, "Reinforcement learning with deep energy-based policies," in ICML, 2017.

[11] W. Zhicheng, W. Wei, A. Xie, Y. Zhang, J. Wu, and Q. Zhu, "Hybrid bipedal locomotion based onreinforcement learning and heuristics," Micro machines, vol. 13, p. 1688, 10 2022.

[12] Q. Nguyen, M. J. Powell, B. Katz, J. D. Carlo, and S. Kim, "Optimized jumping on the mit cheetah 3 r

[13] V. Tsounis, M. Alge, J. Lee, F. Farshidian, and M. Hutter, "Deepgait: Planning and control of quadrupedal gaits using deep reinforcement learning," IEEE Robotics and Automation Letters, vol. 5, pp. 3699–3706, 2020.

[14] S. Ha, J. Kim, and K. Yamane, "Automated deep reinforcement learning environment for hardware of a modular legged robot," in 2018 15th international conference on ubiquitous robots (UR). IEEE, 2018, pp. 348–354.

[15] J. Fu, J. D. Co-Reyes, and S. Levine, "Ex2: Exploration with exemplar models for deep reinforcementlearning," in NIPS, 2017.

[16] T. Xiao, E. Jang, D. Kalashnikov, S. Levine, J. Ibarz, K. Hausman, and A. Herzog, "Thinking while moving: Deep reinforcement learning with concurrent control," ArXiv, vol. abs/2004.06089, 2020.

[17] Z. Xie, P. Clary, J. Dao, P. Morais, J. Hurst, and M. van de Panne, "Learning locomotion skills for cassie:Iterative design and sim-to-real," in CoRL, 2019.

[18] J. Hwangbo, J. Lee, A. Dosovitskiy, D. Bellicoso, V. Tsounis, V. Koltun, and M. Hutter, "Learning agileand dynamic motor skills for legged robots," Science Robotics, vol. 4, 2019.

[19] P. F. Cristiano, J. Leike, T. B. Brown, M. Matric, S. Legg, and D. Amodei, "Deep reinforcement learning from human preferences," in NIPS, 2017.

[20] R. Liu, F. Nageotte, P. Zanne, M. De Mathelin, and B. Dresp, "Deep reinforcement learning for the control of robotic manipulation: A focussed mini-review," Robotics, vol. 10, pp. 1–13, 01 2021.

[21] X. B. Peng, G. Berseth, K. Yin, and M. Van De Panne, "Deeploco: Dynamic locomotion skills using hierarchical deep reinforcement learning," ACM Trans. Graph., vol. 36, no. 4, pp. 41:1–41:13, Jul. 2017. [Online]. Available: http://doi.acm.org/10.1145/3072959.3073602

[22] J. Hwangbo, J. Lee, A. Dosovitskiy, D. Bellicoso, V. Tsounis, V. Koltun, and M. Hutter, "Learning agileand dynamic motor skills for legged robots," Science Robotics, vol. 4, 2019.

[23] S. Fujimoto, D. Meger, and D. Precup, "Off-policy deep reinforcement learning without exploration," in International Conference on Machine Learning. PMLR, 2019, pp. 2052–2062.

[24] T. Hester, M. Quinlan, and P. Stone, "Rtmba: A real-time model-based reinforcement learning architecture for robot control," in 2012 IEEE International Conference on Robotics and Automation, 2012, pp. 85– 90.

[25] A. Hill, A. Raffin, M. Ernestus, A. Gleave, A. Kanervisto, R. Traore, P. Dhariwal, C. Hesse, O. Klimov, A. Nichol, M. Plappert, A. Rad- ford, J. Schulman, S. Sidor, and Y. Wu, "Stable baselines," https://github.com/hill- a/stable-baselines, 2018.obot," 2019 International Conference on Robotics and Automation (ICRA), pp. 7448– 7454, 2019.