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

Contents

List of Figures

List of Tables

Chapter 1

INTRODUCTION

1.1 Robots

A Robots are machines that can be programmed by computers and are capable of performing 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 known as 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[1]

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 humans are 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-level and more sophisticated jobs by combining these robots with AI, going beyond mere automation.

1.2 Robotics

Robotics is the study of the engineering, production, and application of robots.

These days, there is a lot of interest in robots. In our minds, a robot is a device that behaves and looks like a human.In actuality, the definition of a robot is a mechanical

device produced by humans that is capable of autonomous movement, whose motion must be planned, sensed, actuated, and controlled, and whose mobility is controlled by ”programming.” Human senses including objective, touch, and the ability to identify temperature may be shared by robots. Some are even capable of making straightforward decisions.

There are many other kinds of robots, but as our research focuses on legged robots, here are a few examples.

1.2.1 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 robots when 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. Four-legged robots are more common than tripedal robots. They move in pairs while alternating their steps.

Tetrapod robots, sometimes known as quadrupedal robots, are statically stable, particularly when they are not moving. They walk like animals because they have four legs and a comparable gait. They stand with good postural balance. When moving slowly, they can either move one leg at a time to maintain a stable tripod or they can walk by moving the other pair of legs. The TITAN series, WildCat, Cheetah, and the dynamically stable BigDog are examples of four-legged robots.

Working on quadrupedal robot legs is what we'll be doing. However, there is a quadrupedal leg in this project.

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[2]

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 feedback or a punishment. Using RL, it is possible to tackle a specific class of issues, including those in robotics and gaming where 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 punishing desired 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 improves its 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 learning algorithms for this case use dynamic programming techniques (MDP).

The main difference between traditional dynamic programming techniques and reinforcement learning algorithms is 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.

. The illustration that follows shows reinforcement learning in action. An agent

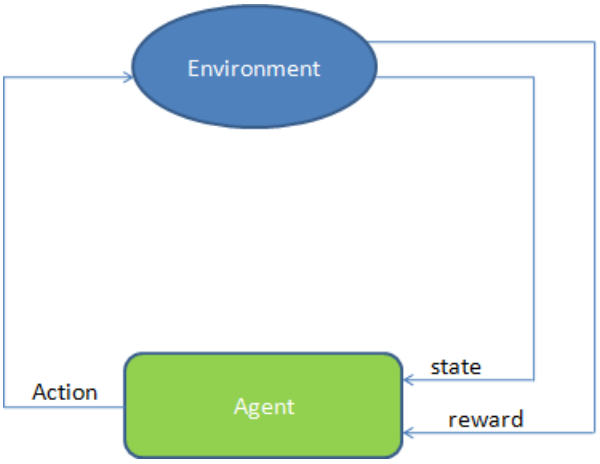

Figure 1.3: reinforcement Learning

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 the set 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 st.

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, angular displacement (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 leg so 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 preset routes. In uncontrolled or free motion, forces govern the shape of the path.

A list of places and relevant times would make a good depiction of the motion of a particle travelling along a straight line. A mathematical equation that depicts position in terms of time would be fundamental for a continuous description.

And here are two types of Kinematics used in robotics to calculate position or angle etc.

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 manipulator is 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 should be 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. An 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 zi axis points along the general manipulator's

rotational or sliding direction.

D-H Parameter:

The four parameters known as Denavit-Hartenberg parameters, or DH parameters, are connected to a specific convention for linking reference frames to the links of a robotmanipulator or spatial kinematic chain.

The following four transformation parameters are known as D–H parameters:

d: offset along previous z to the common normal.D is the distance along J0 Z axis totranslate.

Theta: angle about previous z, from old x to new x.Theta is the angle to rotate aroundthe 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 torotate 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 tothe start of the chain. Inverse kinematics is a mathematical technique that establishesthe 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 translating the 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 trial and error is taken into account. Because deep learning is incorporated into the solution, agents can make judgements based

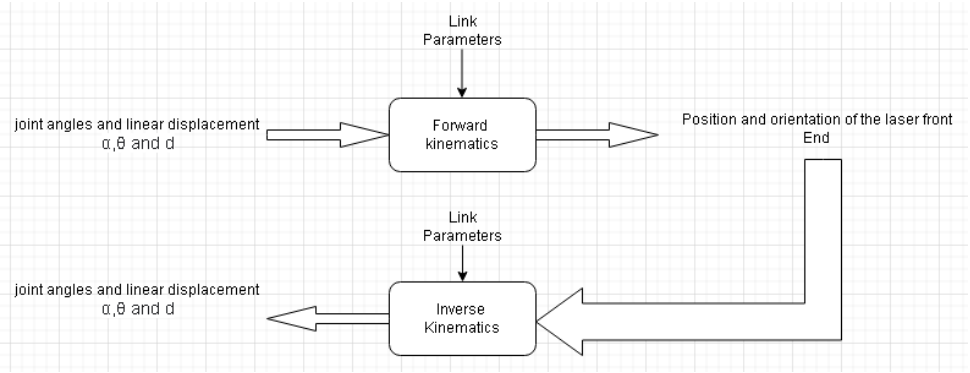

Figure 1.4: Block diagram of forward and inverse kinematics

on input from unstructured data without having to manually engineer the state space. Deep RL algorithms can decide what actions to take to achieve an objective by analysing very massive inputs, such as every pixel presented to the screen in a video game (eg. 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.

A policy that optimises the ”reward function” or other user-provided reinforcement signal that builds up from the immediate rewards is what the agent is supposed to learn through reinforcement learning. In animal psychology, similar processes seem to take place. For instance, biological brains have a programmed tendency to interpret signals like pain and hunger as negative reinforcements and pleasure and food intake as positive reinforcements. Animals may be taught to engage in actions that maximise these benefits in specific situations. Animals may be able to learn through reinforcement, according to this.

1.4.3.1 Action

Through actions that enable state transitions, the Agent can interact with and change its environment. The Agent is respected by the environment for all of their deeds. The plan of action is specified in the strategy.

1.4.3.2 Environment

Everything the Agent might deal with directly or indirectly; everything that isn’t the Agent. The environment shifts as the Agent moves, and each of these shifts is thought of as a state-transition. A reward results from every action the agent takes. a situation in which one agent is present or is surrounded by other agents. The environment is assumed to be stochastic, or essentially random, in RL.

1.4.3.3 Agent

a Reinforcement Learning problem's component that incorporates both learning and acting and tries to maximise the advantages offered by the environment The model you try to build is the Agent, to put it simply. an entity with the ability to perceive, explore, and respond to its environment.

1.4.3.4 Reward system

a quantity that the agent feels as a direct outcome of their interactions with the environment. Because the Agent's goal is to maximise the overall reward it receives over the course of an episode, rewards are what motivate the Agent to act in a desirable manner. All actions produce rewards, which can be broadly categorised into three categories: positive awards that highlight desired actions, negative awards that highlight undesirable actions, and zero awards, which highlight nothing particularly interesting or original the Agent performed. The environment provided input to the agent so it could evaluate how well it had performed.

1.4.3.5 policy

Using policy, a strategy, the agent decides what to do next based on the current circumstances. A policy is a mapping of certain states to the probabilities of selecting different feasible actions in light of those states. An example of a greedy policy is one that produces the action with the highest expected Q-Value for each state.

1.4.3.6 Value

The discount factor expects long-term benefits as opposed to immediate profits. Under the premise that the Agent is in state s and continues to act in accordance with some policy until the end of the episode, the Value function calculates the overall expected reward.

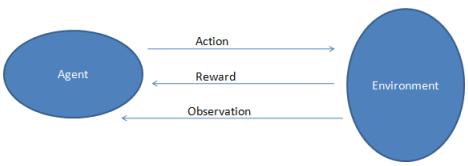

Figure 1.5: Reinforcement learning Working[2]

Chapter 2

Motivation and Objective of Thesis

2.1 Motivation

Humans and animals can be observed moving through their surroundings in a variety of ways, including walking on different surfaces, running, jumping over obstacles, slipping from one location to another, and many more actions they carry out every day. In robotics, such motions are particularly challenging to carry out and perform.

So, how do people and animals learn to do such complex manoeuvres?

Automation and modern technology are advancing to mimic the behaviour and activities of humans and animals.

These movements can be acquired using a variety of techniques and learning processes, but reinforcement learning is a fairly effective strategy. RL agents have the ability to learn on their own.

While certain animal behaviours are inborn, many others are learned via experience. Scientists define learning as an experience-based, relatively long-lasting change in behaviour. Learning usually takes place gradually and in stages.

The kind of behaviours that an animal can learn depend on both its physical constitution and genetic make-up. An animal can only be taught things that they are physically capable of learning. A dolphin cannot learn to ride a bicycle since it has no legs to push the pedals and no fingers to hold the handlebars.

Animals of various species can be seen in these settings. They walk, and their gaits differ from one another. They both sprint and leap, yet they each jump in completely different ways. Like in the illustration above, cats, dogs, kangaroos, springer rats, and many more creatures all respond in quite diverse ways. They differ in terms of kinetics, kinematics, and trajectory. Humans might behave very differently depending on the environment or their gait. The kinematics and dynamics of human jumps, slips, and walks are all distinct from those of animals. Human leg dynamics and trajectory are fundamentally different from those of animals. A man is attempting to jump 50 inches in the illustration above. In various circumstances, their dynamics, such as kinetic energy, velocity, and torque, behave completely differently.

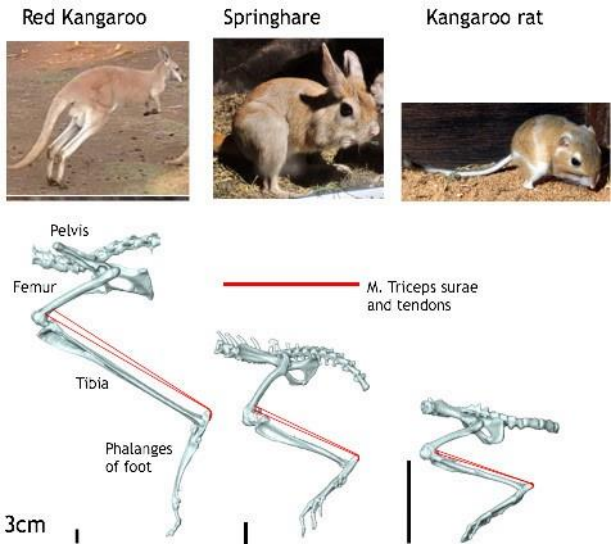

Figure 2.1: Animal jump behavior[3]

Figure 2.2: human jump behavior[4]

So, how can they carry out these kinds of actions in this setting, and how might robots imitate these behaviours in people and other animals? And this serves as the impetus for this undertaking.

2.2 Objective of thesis

1. To observe required torque generation in order to jump on different heights using proprioceptive data using reinforcement learning.

2.3 Literature Review

2.3.1 State of the art

Diego Rodriguez and Suen Behnke 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.[6]. A system utilising deep reinforcement learning is proposed 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.[7]. 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.[8].

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. [9].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.[10]

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.[11]. Ted Xiao, Eric Jang, Dmitry Calashikav, 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.[12].

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. [13] . 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.[14]. A novelty detection system for explorationbased on discriminatively trained exemplar models is described by Fu, John, and Sergey in 2017. a scalable exploration method based on training discriminative exemplar mod- els that shows a novel relationship between exemplar models and density estimates while also awarding novelty bonuses. In order to provide novelty bonuses and show a novel relationship between exemplar models and density estimation, a scalable explo-ration 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 ap- plied in the actual world. Bellman operators that were contemporaneous in continuous and discrete time remained contractions, maintaining the guarantees of Q-learning convergence.[16]

Legged robots can acquire agile movement techniques by imitating real-world animals according to an imitation learning system that Peng, Coumans, 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.[17] 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.

due to extrapolation mistakes that prevent them from learning without data that is connected to the distribution under the current strategy. Batch-constrained deep Q-learning (BCQ) is the first continuous control method that can learn from any batch of data without exploration. Batch-constrained deep Q-learning (BCQ) is the first continuous control method that can learn from any batch of data without exploration. [18]. A model-based RL framework for robot locomotion that achieves walking based on only 4.5 min of data acquired on a quadruped robot is presented by Yang and Caluwaerts et al. in 2019. They discovered that an approach that takes only 4.5 minutes can be achieved by combining accurate long-horizon dynamics learning with multi-step loss functions, careful handling of real-time requirements by accounting for planning latency, and embedding periodicity priors into MPC walking policies.[19] A technique for learning expressive energy-based policies for continuous states and actions, which was previously only achievable in tabular domains, was introduced by Haarnoja and Tang in 2017. Their research can be seen as a soft Q-learning approach with the added benefit of obtaining sophisticated multi-modal policies through approximation inference. Consequently, it was regarded as a type of soft Q-learning that also used approximation inference to build complex multimodal rules. [20]

2.4 Research Gap

1. Vertical jump behavior with RL approach on different heights using proprioceptive data. (research on proprioceptive data has not been reported yet by authors).
2. Vertical jump behavior motion can be generated with single linear policy using proprioceptive data.
3. Required torque generation in order to jump on different heights using proprioceptive data .

Chapter 3

Control and Learning of the Robot

3.1 Control of the robot

3.1.1 Dynamics

We had used two degree of freedom planer robotic leg for analysis purpose shown in figure 2.

there are meaning of all n notations.

q_i = joint angle of the joint i. m_i = mass of link i. l_i = link length of joint i.

The Co-ordinates for knee is:

$x_1 = 0.5l_1 \cos\vartheta_1$, $y_1 = -0.5l_1 \sin\vartheta_1$

we have the forward kinematics equations for the $2_{DOF} legis as follows$:

$x = l_1 \cos\vartheta_1 - 0.5l_2 \sin\vartheta_2$, $y = -l_1 \sin\vartheta_1 - 0.5l_2 \cos\vartheta_2$

We will differentiate both equation of x and y with respect to time t,

$$\dot{x}_1 = -0.5l_1 \sin\vartheta_1\dot{\vartheta}_1$$

$$\dot{y}_1 = -0.5l_1 \cos\vartheta_1\dot{\vartheta}_1$$

$$\dot{x} = -l_1 \sin(\vartheta_1)\dot{\vartheta}_1 - 0.5l_2 \cos(\vartheta_2)\dot{\vartheta}_2$$

$$\dot{y} = -l_1 \cos(\vartheta_1)\dot{\vartheta}_1 + 0.5l_2 \sin(\vartheta_2)\dot{\vartheta}_2$$

so here the velocity for knee and hip of leg is:

$$v_1 = 0.25l_1^2 \sin^2(\vartheta_1)\dot{\vartheta}_1^2 + 0.25l_1^2 \cos^2(\vartheta_1)\dot{\vartheta}_1^2$$

$$v_2 = \left(-l_1 \sin(\vartheta_1)\dot{\vartheta}_1 - 0.5l_2 \cos(\vartheta_2)\dot{\vartheta}_2\right)^2 + \left(-l_1 \cos(\vartheta_1)\dot{\vartheta}_1 + 0.5l_2 \sin(\vartheta_2)\dot{\vartheta}_2\right)^2$$

Center of mass of each link of a leg is right at the middle of the link. And gravitational force act in the positive direction of the y axis.

Here notations are $\vartheta = (\vartheta_1, \vartheta_2)^T$ as the generalized coordinates and here T is the kinetic energy , P is the potential energy and we use Lagrangian Function $L = T - P$.

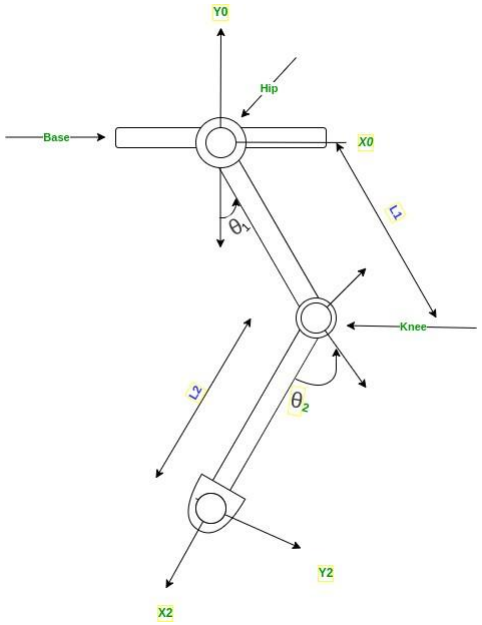

Figure 3.1: A single leg Robot architecture

The Kinetic energy of a leg is written below:

$$T = 0.5I_2\dot{\vartheta}_2^2 + 0.5m_2\left(l_1^2\dot{\vartheta}_1^2 - l_1l_2\cos(\vartheta_1+\vartheta_2)\dot{\vartheta}_1\dot{\vartheta}_2 + 0.25l_2^2\dot{\vartheta}_2^2\right) + \left(0.5I_1 + 0.125l_1^2m_1\right)\dot{\vartheta}_1^2$$

The Potential energy of a leg is :

$$P = -g\left(0.5l_1m_1\sin(\vartheta_1) + m_2\left(l_1\sin(\vartheta_1) + 0.5l_2\cos(\vartheta_2)\right)\right)$$

Using T and P let us derive the equation of motion for this leg are ,

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{\vartheta}_i}\right) - \frac{\partial L}{\partial \vartheta_i} = \tau_i \tag{3.1}$$

where i = 1,2

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{\vartheta}_1}\right) - \frac{\partial L}{\partial \vartheta_1} = \tau_1 \tag{3.2}$$

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{\vartheta}_2}\right) - \frac{\partial L}{\partial \vartheta_2} = \tau_2 \tag{3.3}$$

putting the value of T and P in the formula of L and the equation of dynamics of the Two-link Robotic leg:

$$\tau_1 = I_1\ddot{\vartheta}_1 - 0.5gl_1m_1\cos(\vartheta_1) - gl_1m_2\cos(\vartheta_1) + 0.25l_1^2m_1\ddot{\vartheta}_1 + l_1^2m_2\ddot{\vartheta}_1$$

$$+ 0.5\ l_1 l_2 m_2 \sin(\vartheta_1 + \vartheta_2)\dot{\vartheta}_2^2 - 0.5 l_1 l_2 m_2 \cos(\vartheta_1 + \vartheta_2)\ddot{\vartheta}_2 \quad (3.4)$$

$$\tau_2 = 1.0 I_1 \ddot{\vartheta}_1 - 1.0 I_2 \vartheta_2 + 0.5 g l_2 m_2 \sin(\vartheta_2) + 0.25 l_1^2 m_1 \ddot{\vartheta}_1 + 1.0 l_1^2 m_2 \ddot{\vartheta}_1$$

$$+ 0.5\ l_1 l_2 m_2 \sin(\vartheta_1 + \vartheta_2)\dot{\vartheta}_2^2 - 0.5 l_1 l_2 m_2 \cos(\vartheta_1 + \vartheta_2)\ddot{\vartheta}_2 \quad (3.5)$$

3.2 Model based approach

Systems that use reinforcement learning have two options for decision-making. In the model-based approach, a system asks questions of the type ”what will happen if I do x?” using a predictive model of the environment to choose the optimum x1. The alternative model-free technique completely omits the modelling step in favour of teaching a control strategy from scratch. Although in practise the distinction between these two approaches can become hazy, it serves as a rough guide for segmenting the range of algorithmic alternatives.

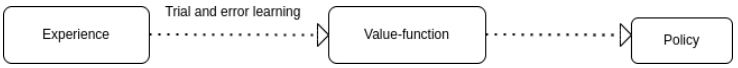

Figure 3.2: Model-free approach

3.3 Reinforcement learning algorithms

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.

3.3.1 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 shall first 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 course of 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, Actor and Critic, is the most popular way to implement PPO.

$L_{clip}(\vartheta) = E_l[min(r_t(\vartheta)A_t, clip(r_t(\vartheta), 1 - \epsilon, 1 + \epsilon)A_t)]$

$E_t = empiricalexpectationovertime - steps.r_t = ratioo\,f\,theprobabilitiesunderthenewandoldpolicy.($ $estimateadvantageattimet.\epsilon = hyper - parameter, usually 0.1 or 0.2.$

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 to improve 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. $\epsilon isahyper - parameterdenotesthelimito\,f\,therangewithinwhichtheupdateisallowed.$

Here are algorithm for PPO.

Input: initial policy parameters.

for iteration = 1, 2, 0, 0, 0. do

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

Run policy $\pi\vartheta_{oLd}inenvironment\,f\,orTtime - stepscomputeadvantageestimateA_1, \ldots\ldots\ldots\ldots, A_i$

$end\,f\,or$

$optimizesurrogateLwithrespecttoϑ, withkepochsandmini - batchthem <= NT$

$\vartheta_{0ld} < - - - - - \vartheta$

end for

What we can see is that small batches of observations, sometimes known as ”mini-batch,” are used for updating and then discarded in order to make room for a fresh batch of data. To prevent large revisions that could potentially be irreparably detrimental, the new policy will be ”-clipped” to a tiny zone. In other words, PPO operates just like other 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.

3.3.2 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 one of the ways the brain picks up new information.

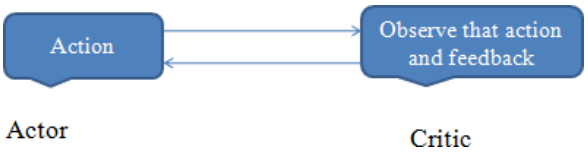

Figure 3.3: A2C Working

As is well known, a neural network can be used to parameterize the Q function and learn the Q value. This brings up 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 for improvement. It can benefit from Q-learning efficiency techniques like memory replay. At each update phase, we update 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 is what we'll refer to as the advantage value:

$A(S_t, a_t) = Q(S_t, a_t) - v(a_t)$

Where ,

$a_t = actiontakenaspertime$

$S_t = Stateofanaction$

and here are the psedocode for A2C algo 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 3.4: Psedocode for A2C[5]

Chapter 4

Results and Discussion

Parameters	value
K epochs	30000
eps clip	0.2
gamma	0.99
lr actor	0.003
lr critic	0.001
time step	0
max ep len	60000
update time-step	max episode length * 4
max training time-steps	le5

Table 4.1: Training parameters

4.1 Results and Discussion

This project aims to teach a robotic limb how to jump over a specified height. Robotic legs may also imitate a variety of motions, including running, slipping, jumping, and walking. With a lot of practice and learning, this is feasible. To train that limb, reinforcement learning is employed. There are many algorithms in reinforcement learning that can facilitate learning and training, but PPO (proximal policy optimizer) is employed in this case.

The goal is to jump higher than a predetermined height, and reinforcement learning is used to achieve this goal. The leg's height was altered for each iteration when the limb was recreated on the MUJOCO simulator. The figure displays the outcomes.

Figure 4.1: Training simulation of the leg in mujoco-py environment.

Chapter 5

Conclusion

After conducting experiments in order to accomplish the goal and analyzed the findings to reach the following conclusions:

1. By employing reinforcement learning, a model-based strategy, and no control system, model will be train to jump beyond a specified height.
2. Reinforcement learning lead us to explore feasible implementation of highly dynamic movements without any equation of motion.
3. Using reinforcement learning we can achieve higher height .

Chapter 6

Future Scope

Future research on robotic legs utilising reinforcement learning will be extensive, and there is potential for learning these models using algorithms.Following this project, these are the primary areas where advancement will be made:

1.In this project, a robotic arm is trained utilising a variety of reinforcement learning techniques using the PPO algorithm.

2. Quadruped models can be trained using reinforcement learning approaches.

Bibliography

[1] Gabe Nelson, Aaron Saunders, and Robert Playter. The petman and atlas robots at boston dynamics. *Humanoid Robotics: A Reference*, 169:186, 2019.

[2] Quang Nguyen, Matthew J. Powell, Benjamin Katz, Jared Di Carlo, and Sangbae Kim. Optimized jumping on the mit cheetah 3 robot. *2019 International Conference on Robotics and Automation (ICRA)*, pages 7448–7454, 2019.

[3] Dondogjamts Batbaatar and Hiroaki Wagatsuma. A proposal of the kinematic model of the horse leg musculoskeletal system by using closed linkages. In *2019 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, pages 869–874. IEEE, 2019.

[4] Dondogjamts Batbaatar and Hiroaki Wagatsuma. A proposal of the kinematic model of the horse leg musculoskeletal system by using closed linkages. In *2019 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, pages 869–874. IEEE, 2019.

[5] Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *International conference on machine learning*, pages 1928–1937. PMLR, 2016.

[6] Diego Rodriguez and Sven Behnke. Deepwalk: Omnidirectional bipedal gait by deep reinforcement learning. *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3033–3039, 2021.

[7] Guillaume Bellegarda and Quan Nguyen. Robust quadruped jumping via deep reinforcement learning. *ArXiv*, abs/2011.07089, 2020.

[8] Julian Ibarz, Jie Tan, Chelsea Finn, Mrinal Kalakrishnan, Peter Pastor, and Sergey Levine. How to train your robot with deep reinforcement learning: lessons we have learned. *The International Journal of Robotics Research*, 40:698 – 721, 2021.

[9] Sehoon Ha, Joohyung Kim, and Katsu Yamane. In *2018 15th International Conference on Ubiquitous Robots (UR)*, pages 348–354. IEEE, 2018.

[10] Tuomas Haarnoja, Aurick Zhou, Sehoon Ha, Jie Tan, G. Tucker, and Sergey Levine. Learning to walk via deep reinforcement learning. *ArXiv*, abs/1812.11103, 2019.

[11] Ted Xiao, Eric Jang, Dmitry Kalashnikov, Sergey Levine, Julian Ibarz, Karol Hausman, and Alexander Herzog. Thinking while moving: Deep reinforcement learning with concurrent control. *ArXiv*, abs/2004.06089, 2020.

[12] Quang Nguyen, Matthew J. Powell, Benjamin Katz, Jared Di Carlo, and Sangbae Kim. Optimized jumping on the mit cheetah 3 robot. *2019 International Conference on Robotics and Automation (ICRA)*, pages 7448–7454, 2019.

[13] Zhaoming Xie, Patrick Clary, Jeremy Dao, Pedro Morais, Jonanthan Hurst, and Michiel van de Panne. Learning locomotion skills for cassie: Iterative design and sim-to-real. In *CoRL*, 2019.

[14] Ted Xiao, Eric Jang, Dmitry Kalashnikov, Sergey Levine, Julian Ibarz, Karol Hausman, and Alexander Herzog. Thinking while moving: Deep reinforcement learning with concurrent control. *ArXiv*, abs/2004.06089, 2020.

[15] Justin Fu, John D. Co-Reyes, and Sergey Levine. Ex2: Exploration with exemplar models for deep reinforcement learning. In *NIPS*, 2017.

[16] Ted Xiao, Eric Jang, Dmitry Kalashnikov, Sergey Levine, Julian Ibarz, Karol Hausman, and Alexander Herzog. Thinking while moving: Deep reinforcement learning with concurrent control. *ArXiv*, abs/2004.06089, 2020.

[17] Xue Bin Peng, Erwin Coumans, Tingnan Zhang, Tsang-Wei Edward Lee, Jie Tan, and Sergey Levine. Learning agile robotic locomotion skills by imitating animals. *ArXiv*, abs/2004.00784, 2020.

[18] Scott Fujimoto, David Meger, and Doina Precup. Off-policy deep reinforcement learning without exploration. In *ICML*, 2019.

[19] Yuxiang Yang, Ken Caluwaerts, Atil Iscen, Tingnan Zhang, Jie Tan, and Vikas Sindhwani. Data efficient reinforcement learning for legged robots. *ArXiv*, abs/1907.03613, 2019.

[20] Tuomas Haarnoja, Haoran Tang, P. Abbeel, and Sergey Levine. Reinforcement learning with deep energy-based policies. In *ICML*, 2017.