






Achieving Accurate Trajectory Control While Training Legged Robots Using Machine Learning

Amit Biswas¹ , Neha N. Chaubey² , and Nirbhay Kumar Chaubey³ 

¹ Triassic Robotics Pvt Ltd., Delhi, India
amit@triassicrobotics.com

² Dharmsinh Desai University, Gujarat, India
nchaubey123@gmail.com

³ Ganpat University, Gujarat, India
nirbhay@ieee.org

Abstract. Legged robots are a class of robotic systems designed to move and navigate using leg mechanisms, similar to how animals with legs move. Legged robots differ from other mobile robots primarily in their mode of locomotion. They can traverse various types of terrain, uneven surfaces and human made structures like stairs, steps etc. They have significant advantages over other forms of robotic locomotion, such as wheels or tracks, but they are complex to build and control. Designing and controlling legged robots can be complex due to the need for stability, balance, and coordination among multiple legs. Building these control mechanisms have always been a challenging task. In this paper we discuss about a four-legged robot that we built and how we trained and controlled its motion. We discuss how we can use latest techniques in Machine Learning (ML) to train locomotion skills to legged robots. We evaluate training directly in the cartesian space as compared to the more popular approach of using joint space. We also look at the challenges we faced in trajectory control and how we solved them. We discuss how we achieved accurate tracking of trajectories and explain the importance of accurate trajectory tracking in legged locomotion. We describe our experimental setup including the simulation environment we used, the tools and techniques used in setting up the experiments and how we built a real robot and used it for our experiments.

Keywords: Legged Robots · Legged locomotion · Motion Control · Trajectory Control · Cartesian Space · Joint Space · Machine Learning · Reinforcement Learning · Augmented Random Search

1 Introduction

Robotic locomotion is a challenging task especially on uneven or unstructured terrains. For legged robots, its even more of a challenge. Compared to other forms of robotic locomotion such as wheel or tracks, Legged locomotion is much more complicated

because of the need for stability, balance, and coordination between multiple legs. Legged locomotion is difficult to achieve because of the complexities involved in building and controlling the mechanisms that enable legged motion.

These complexities can be broadly classified into two categories: (i) Determining and synchronizing the leg movements that are required to achieve a desired motion and (ii) Accurately effecting the desired motion through a series of mechanisms including motor controllers, sensors etc. Achieving a fine balance between these components is also as critical as solving these complexities.

The challenge of legged locomotion also involves processing the large set of possible actions. The control of legged robots is inherently complex due to the coordination of multiple legs and joints. Achieving smooth and efficient locomotion while adapting to varying terrains and obstacles is a significant challenge. Legged robots often require substantial power to move and maintain stability. The mechanical design of legged robots needs to account for both structural integrity and flexibility. The robot's limbs and joints must be robust enough to withstand dynamic movements and external forces while being lightweight to conserve energy. Achieving the right balance between strength, weight, and flexibility can be a challenge. Compared to wheeled or tracked robots, legged locomotion consumes more energy. Efficient energy management and power supply systems are necessary to extend the operational time and range of legged robots, especially in field applications where recharging or refueling may be limited.

In legged robots, the body of the robot is supported by mechanical structures representing legs or limbs. These legs are used for movement of the robot as well. Sequence of leg movements are carefully synchronized to set the body in motion. Leg movements and their placements must be orchestrated to keep the body in motion while maintaining balance and stability. Computing the movement of the legs and then executing the motion in hardware, is the core of the challenge of legged locomotion [1].

The irregular nature of the terrain introduces more complexities in the process. If the surface is not even, support from legs becomes unpredictable. Not all possible positions may be suitable for footfall, and some of the possible positions may not even be achievable due to the uneven nature of the terrain. Sometimes the position determined by the controller may not be reachable due to surface irregularities. These unpredictable elements make planning the leg movements even more difficult. Uneven surface also creates unpredictable reactionary forces on contact. These reactionary forces vary in magnitude and direction and is difficult to factor in while planning leg movements. These challenges, arising from the interaction of the robot to the environment, introduces new set of complications in the walking process [2].

For the first part of the problem, ie. Determining and synchronizing leg movements, various different control mechanisms have been developed in the past. Most of these traditional methods relied heavily on purely physics-based controllers. These controllers are difficult to build and maintain and requires substantial amount of modifications to adapt for any change in the environment. These controllers are not versatile to subtle changes in the environment. A much more feasible solution is required to make legged motion effective. Recent advancements in the field of Artificial Intelligence (AI) and Machine Learning (ML) techniques have opened up new possibilities for controlling motion of legged robots and for training them as well. These new control systems that

are based on Machine Learning, are found to be much more resilient to environmental factors.

Traditional controllers that were purely physics based took a different approach by attempting to control the robot by observing its state and then calculating the necessary changes required. These controllers would measure physical aspects of the robot such as joint positions, body state etc. and use these values to calculate a possible state for the legs. This approach can be further advanced to achieve complex movements such as walking. Similarly balance and stability can also be achieved while walking. These methods of controlling the robot worked well under ideal conditions but even minor changes in the environment would create problems. Small deviations in environment often give rise to conditions that are the controller cannot handle. Under challenging environments, these controllers fail to perform. In addition to the fragile nature of these controllers, they are also difficult to make. They require deep knowledge across multiple domains to understand and develop the control policies. After development they need a lot of fine-tuning as well.

Artificial intelligence and Machine Learning techniques have started playing a crucial role in controlling legged robots, enabling them to adapt, learn, and make autonomous decisions in complex environments. Algorithms based on machine learning techniques are used to plan and control the motions of legged robots. These algorithms learn from experience or simulate the robot's movements to find suitable control strategies that maximize stability, energy efficiency, and locomotion performance. By observing and imitating human movements or predefined motion sequences, legged robots can learn how to navigate and perform specific tasks. These algorithms allow legged robots to adapt their locomotion strategies in real-time based on sensor feedback. Reinforcement learning or adaptive control algorithms can adjust leg trajectories, foot placement, or gait patterns to optimize stability, maintain balance, and improve performance in response to changing terrain or external disturbances. This approach of controlling the robot makes it much more resilient as compared to traditional controllers.

The second part of the challenge is to accurately execute the desired leg movements. Accurate trajectory control becomes very necessary here. In-order to have accurate tracing of trajectories, several things have to be synchronized such as the motor controllers, the feedback sensors, the control algorithms etc. We found that minor discrepancies sensor feedback or motor control often leads to major deviations in the tracked trajectory. Under certain conditions such as at high velocities, the deviations are even more magnified. Higher inertia of the moving part also contributes to increased deviations and jittery movements. Maintaining the desired velocity and acceleration throughout the trajectory is often a challenge because of the discrepancies in the feedback or control loop as well. We found a solution by leading the positional value of the trajectory by a few steps ahead so that the acceleration and velocity can remain smooth and need not change abruptly at each step of control.

2 Related Work

Much research has been done in the field of robotic locomotion. Many different methods have been proposed and evaluated. The Cheetah robot, developed by MIT researchers, works by generating simple reference trajectories. In order to get desired contact forces,

it performs model predictive control and then uses Jacobian transpose control to realize them [3]. The ANYmal robot [4] uses the inverted pendulum model to plan for footholds [3]. These control algorithms work well but require deep knowledge across multiple domains including the dynamics of the robot, physical aspects of the robot and its capabilities. Often these requirements become a limiting factor. In contrast, AI based control mechanisms can be used for training even without any prior know-how of the dynamics of the robot or its physical aspects. This can be achieved through Reinforcement Learning techniques.

Literature survey shows several instances where machine learning techniques have been used for robotic locomotion [5–8]. Multiple approaches have been proposed based on reinforcement learning techniques. They results vary on performance and success levels. Reinforcement Learning can be used to explore different leg trajectories and coordination patterns, these algorithms can discover gaits that minimize energy consumption, maximize stability, or adapt to varying terrains and are found to be much better than traditional methods of control [8]. Robots can be trained in simulation using techniques like Deep Reinforcement Learning. Simulations provide a valuable platform for training and optimizing legged robot controllers. AI-based algorithms can be applied to train legged robots in simulated environments, leveraging reinforcement learning. Simulations allow for faster and safer exploration of control policies, enabling legged robots to learn and improve their locomotion capabilities before deployment in the real world. However, simulation differs from the real world in several factors, and all of these factors cannot be accurately modelled in simulation. This creates discrepancies between the real world and the simulated environment.

The discrepancies between simulation and the real world create a gap that is sometimes referred to as the sim-to-real gap. Numerous solutions have been proposed to overcome this sim-to-real gap [9]. An easy solution would be to use a physical setting that represents the real-world setup very closely and train the robot in this environment. This approach effectively removes any sim-to-real gap because simulation has been removed from the process altogether. In most scenarios this approach is not feasible because of multiple factors such as it is not possible to collect training data from the physical environment, changing the robot’s dynamics is now possible, and its time consuming to iterate. Using simulations for training can be a much more efficient solution. We can conduct the training process completely in a simulated environment and on successful training, we can transfer the trained policies to a hardware robot.

In our setup, we will evaluate a relatively newly proposed algorithm called Augmented Random Search (ARS) [10] that is claimed to be much more efficient than many existing algorithms. According to the algorithm’s authors, ARS beats the fastest competing methods by about 15 times [10].

While using Machine Learning to train legged robots, selection of action space has large effect on both training time and quality of trained policies [11]. Most of the previous work has focused on learning in joint space. However, cartesian space can be an equally good action space and, in some studies, cartesian space control has been found to be better than joint space [11].

Most of the previous work does not focus on the challenges of executing the learned motion on real hardware. The trained policies were only validated on simulated environments. However, it is tricky process to transfer trained policies from a simulated environment to real-world hardware [9].

3 Using Machine Learning to Train Locomotion Policies

Machine Learning has emerged as the most successful method of training complex motion control behavior in robots [12]. Using Machine Learning we can virtually eliminate the need of any manual tuning or precise fine tuning of the control parameters. This is a great benefit especially for complex motion control systems like in legged robots because the system can be made to learn the intricacies by itself instead of being programmed by a human. Reinforcement learning (RL) is often used for training legged robots due to its ability to learn complex behaviors and adapt to dynamic environments. Reinforcement learning enables legged robots to learn through trial and error. The robot explores its environment, takes actions, and receives feedback or rewards based on the outcomes. RL algorithms learn from these experiences and adjust the robot's behavior to maximize the cumulative reward over time. This iterative learning process allows legged robots to discover effective locomotion strategies and adapt to different terrains or tasks. Legged robots often undergo initial training in simulated environments before being deployed in the real world. RL is well-suited for simulated training as it enables legged robots to explore and learn in virtual environments without the risks or costs associated with real-world trials. Once a legged robot has learned locomotion skills in one setting, RL algorithms can help generalize and adapt those skills to new situations.

In recent research work, a lot of progress has been made in the field of Deep Reinforcement Learning [13–15]. RL algorithms, such as deep reinforcement learning, can learn hierarchical representations, allowing legged robots to generate complex and coordinated movements. Controlling legged robots is a challenging task due to the high-dimensional action spaces and the need for coordination among multiple limbs. Reinforcement learning can handle the complexity of learning control policies by searching and optimizing in large action spaces. In order to eliminate any discrepancies, we try to create a representation of the model as accurately as possible.

We evaluated two approaches while training our robot: first we used joint space by getting joint positions from position sensors and mapping them to the trajectory controller, and second approach is to map the position of the end points of the legs directly in the cartesian space. We observed that using cartesian space can be as effective as using joint space [16]. Whether joint space is used or cartesian space, accurate trajectory control was a crucial factor that determines performance (Fig. 1).

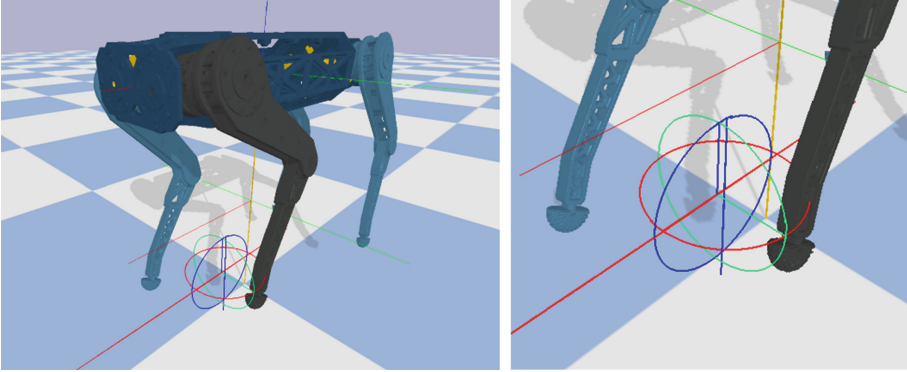


Fig. 1. Tracking trajectories in cartesian space

3.1 Learning in Cartesian Space

In our experimental setup, we trained our robot directly in the cartesian space. Joint space control is popularly used for such training, but cartesian space is also being studied in several studies [17]. We decided to use cartesian space because it was more closely related to real world mapping. A policy trained in cartesian space attempts to control the movements by reading and controlling the cartesian position of the end effector of each leg. This method of controlling, as compared to joint space, has some benefits such as higher sample efficiency and easier transfer between multiple simulation environments. We also found that changing the physical dimensions of the robots leg had less effect while using cartesian space than while using joint space.

Another advantage of cartesian space control is improved tracking of trajectories. We found that without using special techniques that maintain the end effector position at each step, joint space control did not track the trajectories as accurately as cartesian space control did. This is because joint space control does not guarantee uniform amount of movement for all joints in motion although the end result was reached. With cartesian space control, the same end result was reached, but it guaranteed that every step in the trajectory is executed while the end effector remains placed in the trajectory [18]. This results into very accurate tracking performance.

4 Using Augmented Random Search for Training

Legged robots often operate in dynamic and uncertain environments. They need to make real-time decisions regarding gait selection, obstacle avoidance, and navigation. Highly dynamic maneuvers such as walking, running, jumping etc. require careful balancing and stability and are extremely difficult for robots to execute [9]. Developing efficient planning algorithms that can handle the complexity of legged locomotion while considering real-time sensory information is a significant challenge. Numerous applications of machine learning (ML) based techniques have been suggested to address these issues[19]. In our experiments we evaluate a relatively new algorithm named ‘‘Augmented Random Search’’ [10] to train basic locomotion skills such as walking to our custom built quadruped robot.

Augmented Random Search (ARS) is a reinforcement learning algorithm that combines elements of random search and policy gradient methods. It is a simple yet effective algorithm for training static, linear policies for continuous control problems [20]. ARS is an improvement over Basic Random Search (BRS) (Fig. 2).

Algorithm 1 Augmented Random Search (ARS): four versions **V1**, **V1-t**, **V2** and **V2-t**

- 1: **Hyperparameters:** step-size α , number of directions sampled per iteration N , standard deviation of the exploration noise ν , number of top-performing directions to use b ($b < N$ is allowed only for **V1-t** and **V2-t**)
- 2: **Initialize:** $M_0 = \mathbf{0} \in \mathbb{R}^{p \times n}$, $\mu_0 = \mathbf{0} \in \mathbb{R}^n$, and $\Sigma_0 = \mathbf{I}_n \in \mathbb{R}^{n \times n}$, $j = 0$.
- 3: **while** ending condition not satisfied **do**
- 4: Sample $\delta_1, \delta_2, \dots, \delta_N$ in $\mathbb{R}^{p \times n}$ with i.i.d. standard normal entries.
- 5: Collect $2N$ rollouts of horizon H and their corresponding rewards using the $2N$ policies

$$\begin{aligned} \mathbf{V1}: \quad & \begin{cases} \pi_{j,k,+}(x) = (M_j + \nu\delta_k)x \\ \pi_{j,k,-}(x) = (M_j - \nu\delta_k)x \end{cases} \\ \mathbf{V2}: \quad & \begin{cases} \pi_{j,k,+}(x) = (M_j + \nu\delta_k) \text{diag}(\Sigma_j)^{-1/2} (x - \mu_j) \\ \pi_{j,k,-}(x) = (M_j - \nu\delta_k) \text{diag}(\Sigma_j)^{-1/2} (x - \mu_j) \end{cases} \end{aligned}$$

- for $k \in \{1, 2, \dots, N\}$.
- 6: Sort the directions δ_k by $\max\{r(\pi_{j,k,+}), r(\pi_{j,k,-})\}$; denote by $\delta_{(k)}$ the k -th largest direction, and by $\pi_{j,(k),+}$ and $\pi_{j,(k),-}$ the corresponding policies.
- 7: Make the update step:

$$M_{j+1} = M_j + \frac{\alpha}{b\sigma_R} \sum_{k=1}^b [r(\pi_{j,(k),+}) - r(\pi_{j,(k),-})] \delta_{(k)},$$

- where σ_R is the standard deviation of the $2b$ rewards used in the update step.
 - 8: **V2** : Set μ_{j+1} , Σ_{j+1} to be the mean and covariance of the $2NH(j+1)$ states encountered from the start of training^[2]
 - 9: $j \leftarrow j + 1$
 - 10: **end while**
-

Fig. 2. Algorithm of Augmented Random Search [11]

5 Accurate Trajectory Control

Accurate trajectory control plays a fundamental role in achieving desired leg movements. Planning for joint trajectories may be achieved during the training process but it is responsibility of the controller to make sure that the planned trajectories are tracked accurately [21]. The accuracy achieved during training can be easily offset by inaccurate tracking.

In absence of appropriate trajectory control mechanisms in place, significant amount of tracking error was observed. Several factors contributed to the error including physical aspects such as the momentum of the leg or limitations of control accuracy, responsiveness of the motor and limitations in reading positional feedback. Accurate trajectory control allows robots to achieve precise movements and positions.

Inaccuracies in tracking the trajectory arises out of several factors including minor discrepancies in the control-feedback loop [22]. Delays in reading positions, or delays in motor controller contribute to tracking errors. However, one factor that contributed substantially was flaws in the trajectory control method. We noticed that at higher velocities and if mass of moving part is high, the momentum of the moving body becomes large enough to introduce substantial amount of tracking errors. The controller attempts to correct this by compensating the values of acceleration and velocity. This ends up being over compensated in the opposite direction and the process repeats again (Fig. 3).

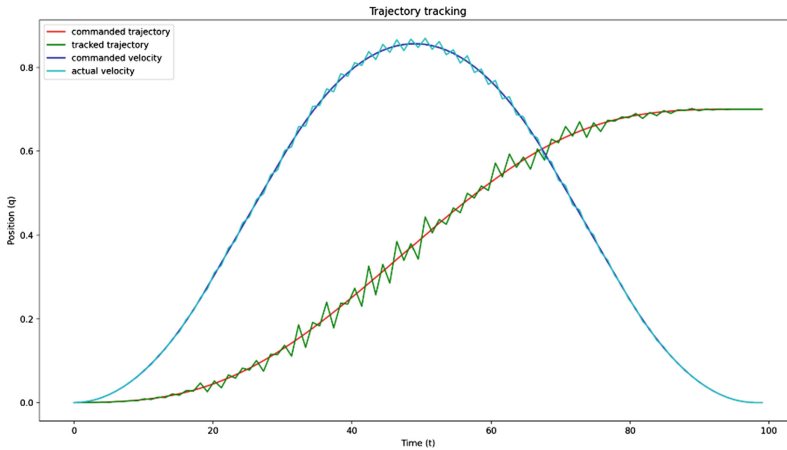


Fig. 3. Poor tracking of trajectories, jittery and abrupt movements

5.1 Using Positional Lead in Trajectory Control

We tested a new approach to solve the tracking errors. Out of the three parameters that are maintained during trajectory control, Acceleration, Velocity, and Position, we found that if the position value is led by n steps ahead, it resulted in better tracking performance [1]. The leading factor, n , starts with 0 at the beginning of the trajectory, reaches its maximum value at the middle of the trajectory and drops back to zero at the end of the trajectory. The maximum value is determined experimentally and requires some tuning (Figs. 4 and 5).

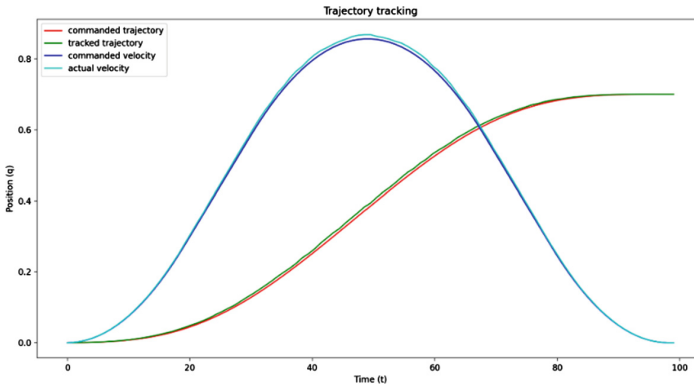
Algorithm 2: Trajectory Control with positional lead in every step

```

//Parameters:
pos_lead      //Factor of positional lead
s_list        //List of all steps in the trajectory
step          //One step in the list of steps s_list
acc           //Acceleration value at the current step
vel           //Velocity value at the current step
pos           //Position value at the current step

While s_list has steps, loop through each step
  step = current step
  acc = step.acc
  vel = step.vel
  if current step is less than total steps - pos_lead
    pos = (current step + pos_lead).pos
  else
    pos = step.pos
  Update motor controller(acc, vel, pos)
  state = Read state from motor controller
  update local state
End while

```

Fig. 4. Positional lead used while tracking trajectories**Fig. 5.** Improved trajectory tracking and smooth movements due to positional lead

6 Experimental Setup

For our experimental setup, we built a real quadruped robot and also setup a simulation environment. We used the real robot to run experiments related to trajectory planning, testing tracking accuracy, and testing motion control. We also took measurements from the robot to get actual real-world values and compared them to our simulation results. Hidden discrepancies that are sometimes not encountered in simulation are often uncovered while testing on real hardware.

We built a fully articulated 4-legged robot. Each leg has 3 degrees of freedom. The joints are referred to as ‘Abad’ joint, the ‘Femur’ joint and the ‘Tibia’ joint. Each joint is powered by an electric motor that drives an actuator. The actuator consists of a set of gear reducers. A high resolution position sensor is attached to each joint, this provides accurate positional feedback to the motor controller (Figs. 6 and 7).



Fig. 6. Leg configuration of “Stego” the real quadrupedal robot built for our experiments

There is an onboard motor controller that receives positional feedback from the high-resolution position sensors connected to each joint. The motor controller uses this data to compute and control the velocities and joint angles. For this experimental setup, we used FOC based motor controllers.

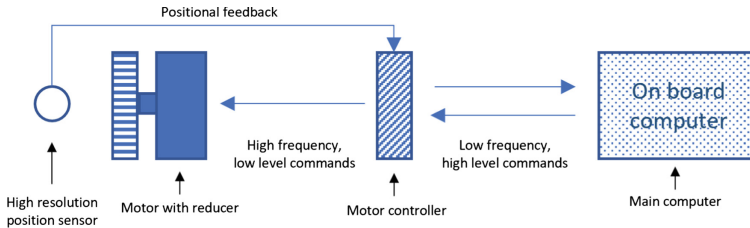


Fig. 7. Joint control and feedback loop

We built a rig to support the leg while testing. The robot leg is mounted on the rig in such a way that it is supported fully but allows vertical movement along the z axis and rotational movement for Femur and Tibia. This setup is connected to a simulated environment on a computer and can be controlled from there (Fig. 8).

6.1 Simulation Environment Setup

We built a simulation environment that represents real-world scenarios as closely as possible. We used this setup to test locomotion policies and to try out training sessions as well. On Windows machine, we used PyBullet for the simulation. We built an identical environment on a Ubuntu system using Gazebo for simulation. For modeling the robot, original CAD files were used that was created for making the real robot. URDF models were created out of these robot models and then it was used in simulation (Figs. 9 and 10).

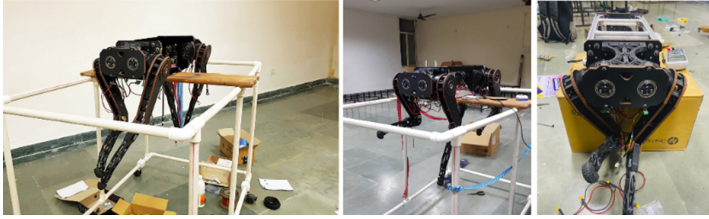


Fig. 8. Stego - The quadruped robot we built to experiment on real hardware



Fig. 9. The real robot leg mounted on the test rig and the simulation environment using PyBullet

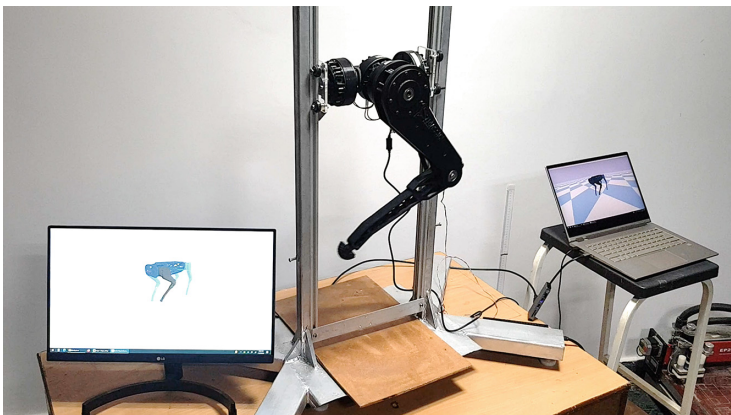


Fig. 10. A single robot leg controlled through the simulation environment

7 Results

We successfully used reinforcement learning methods to train a legged robot basic locomotion skills such as walking. The trained robot successfully executed simple gait patterns such as in walking. We achieved this without requiring deep understanding of the robots. We were able to train the robot without prior domain knowledge. The simulation environment we used to train mimics real-world scenarios very closely. On completion of the training process, the robot was able to execute the trained policies such as walking gaits.

We also demonstrate the benefits of accurate trajectory control. In our experiments, we achieved very accurate trajectory tracking both in the real robot leg as well as in simulation. We demonstrated the benefits of using positional lead in trajectory control. With positional lead the controller continuously tries to reach the leading position while acceleration and velocity remain at the current step. By doing so, the controller does try to overcompensate in opposite direction. This results in much better tracking of trajectories and smoother movements.

We validated that training can be done directly in cartesian space as compared to the more common approach of using joint space. Using cartesian space has some benefits such as improved tracking of trajectories, better transferability between simulation environments and more resilient against any changes in the robots hardware.

We successfully built a fully articulated quadrupedal legged robot. We used this robot for running experiments on real hardware. We establish that controlling legged motion can be achieved by training in simulation and transferring the trained policies to real hardware. Several challenges remain while transferring policies from simulation to real hardware (Figs. 11 and 12).

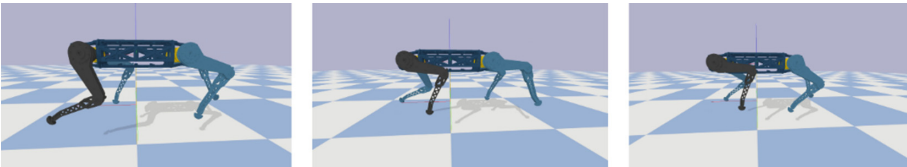


Fig. 11. Execution of walking gait in simulation

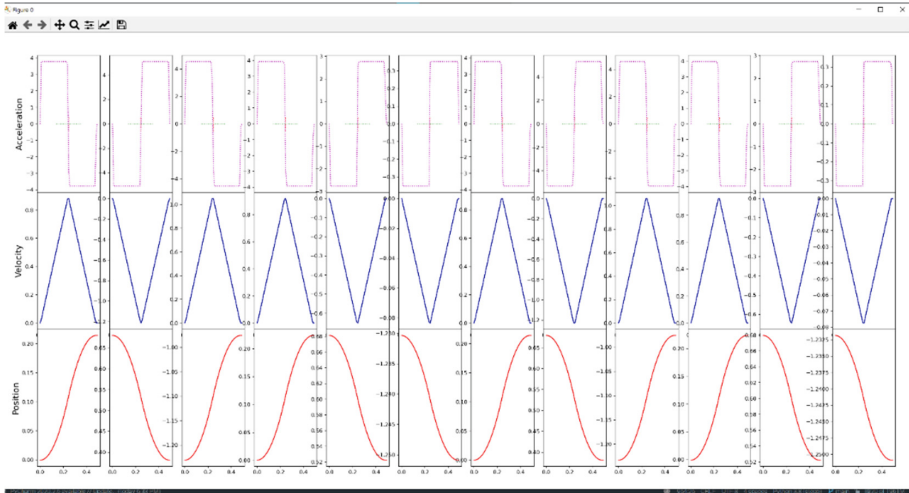


Fig. 12. Simultaneous planning of acceleration, velocity and position for all legs of the robot

8 Conclusion

Reinforcement learning offers a flexible and adaptive approach to training legged robots, enabling them to learn complex locomotion behaviors, adapt to dynamic environments, and generalize across tasks. Machine Learning allows legged robots to acquire robust and efficient locomotion skills that enhance their autonomy and versatility in real-world applications. Using Machine Learning we were successful in achieving complex dynamic gaits in simulation that would otherwise be difficult to achieve with any other approach. We evaluated controlling the joints by addressing them in joint space as well directly in cartesian space and we found that cartesian space motion control and learning is a viable alternative to joint space control. Under some circumstances, cartesian space control may be better suited than Joint space control. We found that accurate trajectory control is absolutely crucial in executing legged locomotion. Using positional lead while tracking trajectories improves tracking performance significantly. For future research we want to evaluate fixed-time trajectories in joint space and compare the performance with cartesian space. For trajectory control we want to verify if acceleration and velocity can be led along with position. This may improve tracking accuracy even more than what we achieved with positional lead. Transferring trained policies from the simulation environment to the real robot remains a challenge to be further explored in future.

References

1. Biswas, A., et al.: Training a legged robot to walk using machine learning and trajectory control for high positional accuracy. In: Kautish, S., et al., AI-Enabled Social Robotics in Human Care Services, IGI Global, pp. 172–187 (2023). <https://doi.org/10.4018/978-1-6684-8171-4.ch006>

2. Bledt, G., Powell, M.J., Katz, B., Di Carlo, J., Wensing, P.M., Kim, S.: MIT cheetah 3: design and control of a robust, dynamic quadruped robot. In: IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Madrid, Spain (2018)
3. Haarnoja, T., Ha, S., Zhou, A., Tan, J., Tucker, G., Levine, S.: Learning to Walk via Deep Reinforcement Learning. [arXiv:1812.11103v3](https://arxiv.org/abs/1812.11103v3) (2019)
4. Hutter, M., et al.: Anymal-a highly mobile and dynamic quadrupedal robot. In: International Conference on Intelligent Robots and Systems (IROS), pp. 38–44. IEEE (2016)
5. Kohl, N., Stone, P.: Policy gradient reinforcement learning for fast quadrupedal locomotion. In: IEEE International Conference on Robotics and Automation. ICRA 2004. 2004, vol. 3, pp. 2619–2624. IEEE (2004)
6. Kumar, A., Fu, Z., Pathak, D., Malik, J.: Rma: Rapid motor adaptation for legged robots. [arXiv preprint arXiv:2107.04034](https://arxiv.org/abs/2107.04034) (2021)
7. Haarnoja, T., Ha, S., Zhou, A., Tan, J., Tucker, G., Levine, S.: Learning to walk via deep reinforcement learning. [arXiv preprint arXiv:1812.11103](https://arxiv.org/abs/1812.11103) (2018)
8. Hafner, R., et al.: Towards general and autonomous learning of core skills: a case study in locomotion. [arXiv preprint arXiv:2008.12228](https://arxiv.org/abs/2008.12228) (2020)
9. DeFazio, D., Zhang, S.: Leveraging Human Knowledge to Learn Quadruped Locomotion Policies. [arXiv:2107.10969v2](https://arxiv.org/abs/2107.10969v2) (2021)
10. Tan, J., et al.: Sim-to-Real: Learning Agile Locomotion For Quadruped Robots. [arXiv:1804.10332v2](https://arxiv.org/abs/1804.10332v2) (2018)
11. Mania, H., Guy, A., Recht, B.: Simple random search provides a competitive approach to reinforcement learning. [arXiv:1803.07055](https://arxiv.org/abs/1803.07055) (2018)
12. Bellegarda, G., Chen, Y., Liu, Z., Nguyen, Q.: Robust High-speed Running for Quadruped Robots via Deep Reinforcement Learning. [arXiv:2103.06484](https://arxiv.org/abs/2103.06484) (2021)
13. Bellicoso, C.D., Bjelonic, M., Wellhausen, L., et al.: Advances in real-world applications for legged robots. *J. Field Rob.* **35**, 1311–1326 (2018). <https://doi.org/10.1002/rob.21839>
14. Lillicrap, T.P., et al.: Continuous control with deep reinforcement learning. [arXiv preprint arXiv:1509.02971](https://arxiv.org/abs/1509.02971) (2015)
15. Schulman, J., Wolski, F., Dhariwal, P., Radford, A., Klimov, O.: Proximal policy optimization algorithms. *CoRR*, abs/1707.06347 (2017)
16. Duan, Y., Chen, X., Houthoofd, R., Schulman, J., Abbeel, P.: Benchmarking deep reinforcement learning for continuous control. In: Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48, ICML 2016, pp. 1329–1338. JMLR.org (2016)
17. Bellegarda, G., Nguyen, Q.: Robust Quadruped Jumping via Deep Reinforcement Learning. [arXiv:2011.07089](https://arxiv.org/abs/2011.07089) (2020)
18. Xu, X., Chen, Y.: A method for trajectory planning of robot manipulators in Cartesian space. In: Proceedings of the 3rd World Congress on Intelligent Control and Automation (Cat. No.00EX393), Hefei, vol. 2, pp. 1220–1225 (2000). <https://doi.org/10.1109/WCICA.2000.863437>
19. Kaspar, M., Muñoz Osorio, J.D., Bock, J.: Sim2Real transfer for reinforcement learning without dynamics randomization. In: IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 4383–4388 (2020). <https://doi.org/10.1109/IROS45743.2020.9341260>
20. Smith, L., Kew, J.C., Peng, X.B., Ha, S., Tan, J., Levine, S.: Legged robots that keep on learning: Fine-tuning locomotion policies in the real world. [arXiv preprint arXiv:2110.05457](https://arxiv.org/abs/2110.05457) (2021)

21. Tirumala, S., et al.: Gait Library Synthesis for Quadruped Robots via Augmented Random Search. [arXiv:1912.12907v1](https://arxiv.org/abs/1912.12907v1) (2019)
22. Carpentier, J., Wieber, P.B.: Recent progress in legged robots locomotion control. *Curr. Rob. Rep.* **2**, 231–238 (2021). <https://doi.org/10.1007/s43154-021-00059-0>(2021)
23. Arm, P., et al.: SpaceBok: a dynamic legged robot for space exploration. In: 2019 International Conference on Robotics and Automation (ICRA), Montreal, QC, Canada, pp. 6288–6294 (2019). <https://doi.org/10.1109/ICRA.2019.8794136>