Learning Agile and Dynamic Motor Skills for Legged Robots

Authors - Hwangbo et al.

Presenter: Jahnavi Malagavalli

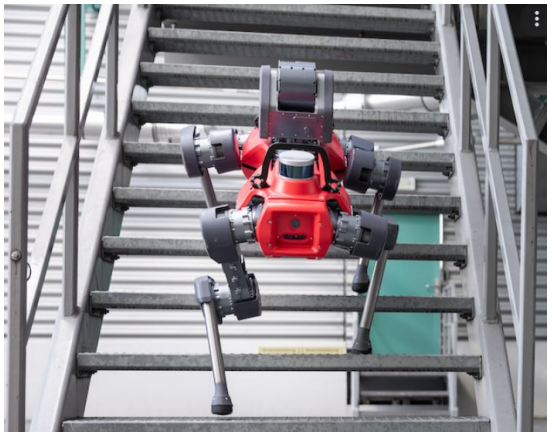
11/03/2022

Motivation

Explore plants and underground tunnels

Source:NASA-JPL/Caltech

Climb stairs



Source:Anybotics

We prefer legged robots compared to tracked/ wheeled ones

- Has freedom to choose contact points.
- Overcome obstacles comparable to their leg length

Main Problem

- Conventional control theories - Insufficient to deal
  - Complex sensors, noise and delays because of information transfer.
  - High-dimensional and non-smooth systems with many physical constraints

- Model and control the behavior of Actuator
  - Existing models do not consider generalization and efficiency

- The proposed work addresses these problems
  - With RL technique to generate a policy for the controller.

Problem Setting

- **Generate a control method for multi-legged robot - ANYmal**
- Combining simulation model and deep reinforcement learning.

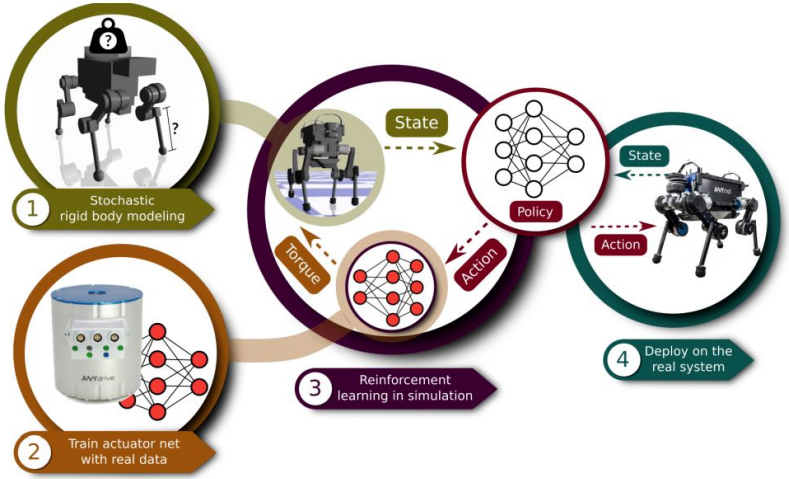


Fig. 1. Creating a control policy. In the first step, we identify the physical parameters of the robot and estimate uncertainties in the identification. In the second step, we train an actuator net that models complex actuator/software dynamics. In the third step, we train a control policy using the models produced in the first two steps. In the fourth step, we deploy the trained policy directly on the physical system.

Context / Related Work / Limitations of Prior Work

Control multi-legged robot as a combination of modules

- Next foothold position.
- Parameterized trajectory
- Trajectory Tracking.

Disadvantage -

- Modeling inaccuracy causes in control inaccuracies
- The design of modular controllers is laborious

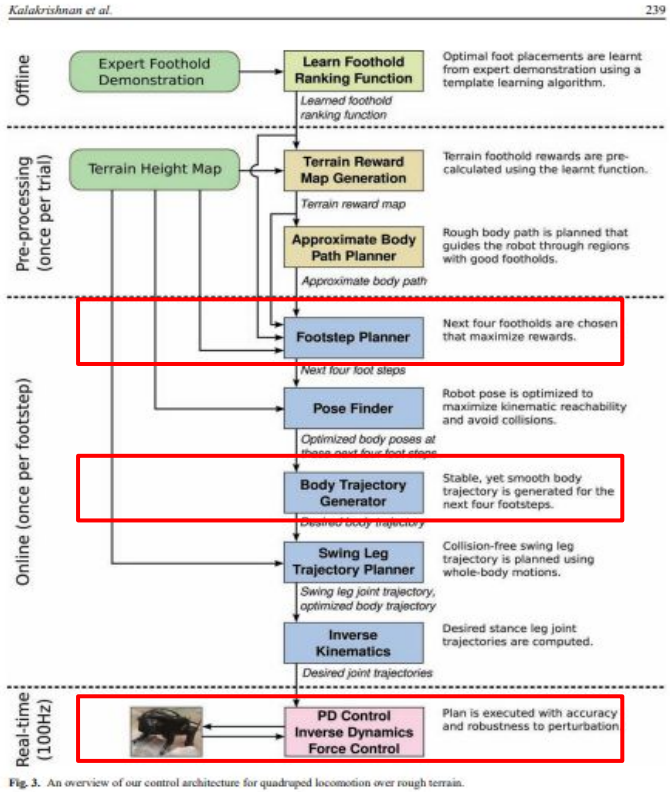


Fig. 3. An overview of our control architecture for quadruped locomotion over rough terrain.

Context / Related Work / Limitations of Prior Work

Control by trajectory optimization -

- Control using two modules of planning and tracking

Disadvantage -

- Parameter tuning to optimize trajectory,it is cumbersome and may fall into local solutions
- The calculation of trajectory optimization is heavy and not suitable for controlling the robot in real time

Context / Related Work / Limitations of Prior Work

Control with existing Reinforcement Learning techniques -

- Use RL to find the optimal control policy

Disadvantage -

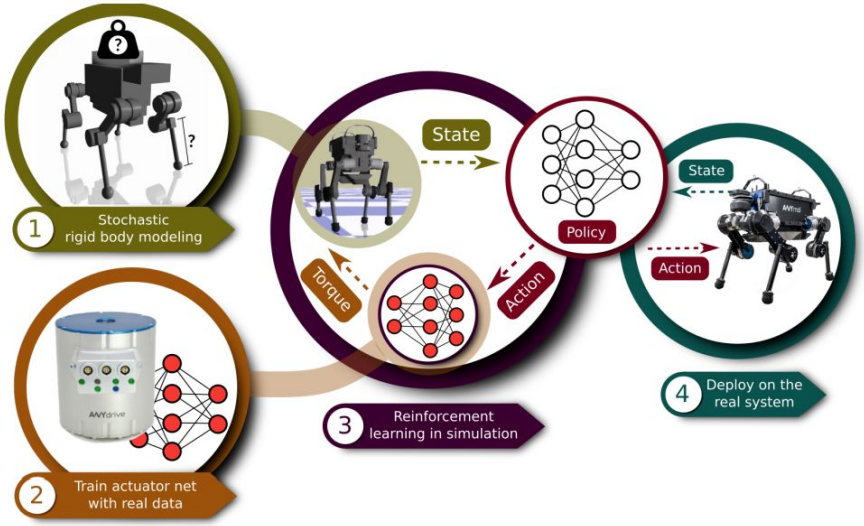
- Sim to Real transfer is hindered by reality gap
- Relatively simple and stable platforms

Proposed Approach / Algorithm / Method

The policy is learned via RL (TRPO) with stimulator-

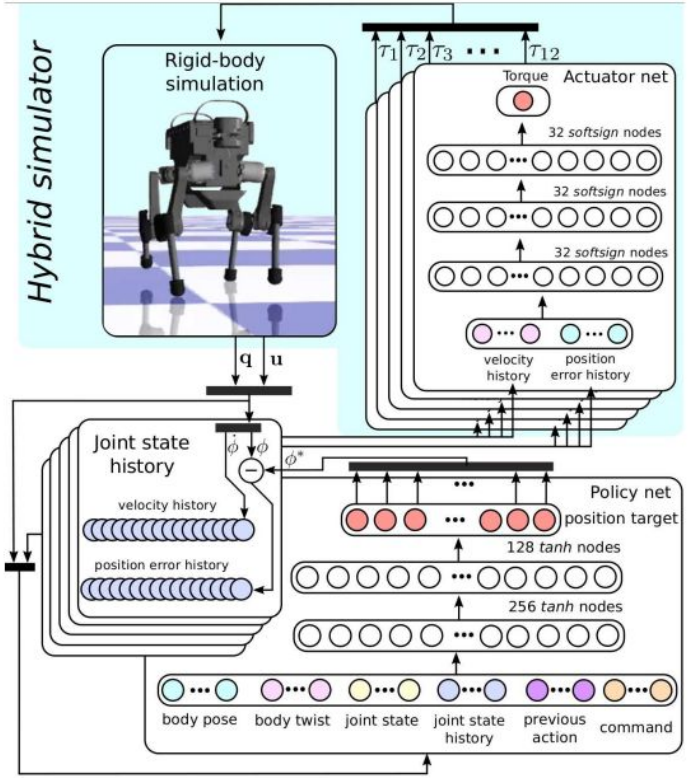
- The policy receives the state observations and outputs the action (joint position) to the actuator

- Learn the relationship between action and torque of the real word actuators with NN



Proposed Approach / Algorithm / Method

- Policy -> maximises the discounted sum of rewards
  - $$\pi^* = \underset{\pi}{\operatorname{argmax}}\, \mathbb{E}_{\tau(\pi)} \left[\sum_{t=0}^{\infty} \gamma^t r_t \right]$$
- q - current coordinates, u - velocity

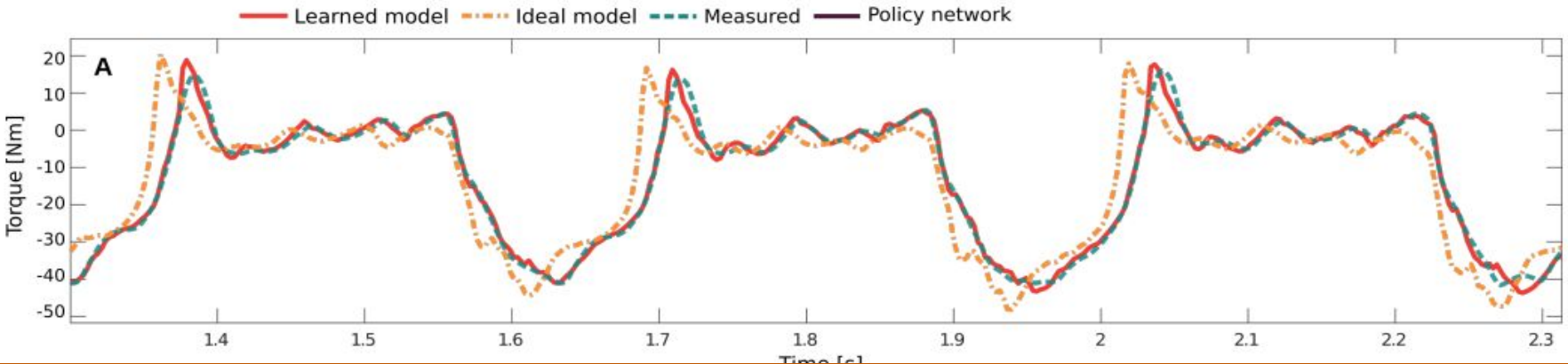


Accuracy of the Actuator Net

Data is collected from 12 actuators on the robot by varying the frequency and amplitude of the trajectory

- MLP Model trained with
  - Input - history of positions errors (generated by a simple controller) and joint velocities
  - Output - joint torques

Comparison with numeric solutions assuming the ideal model

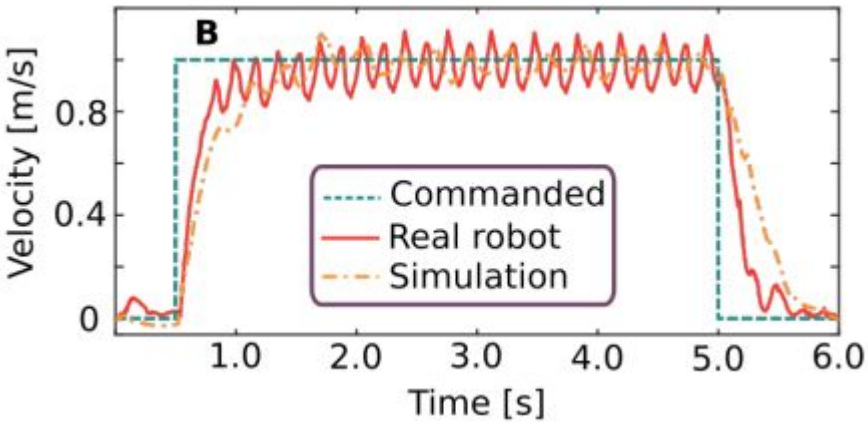


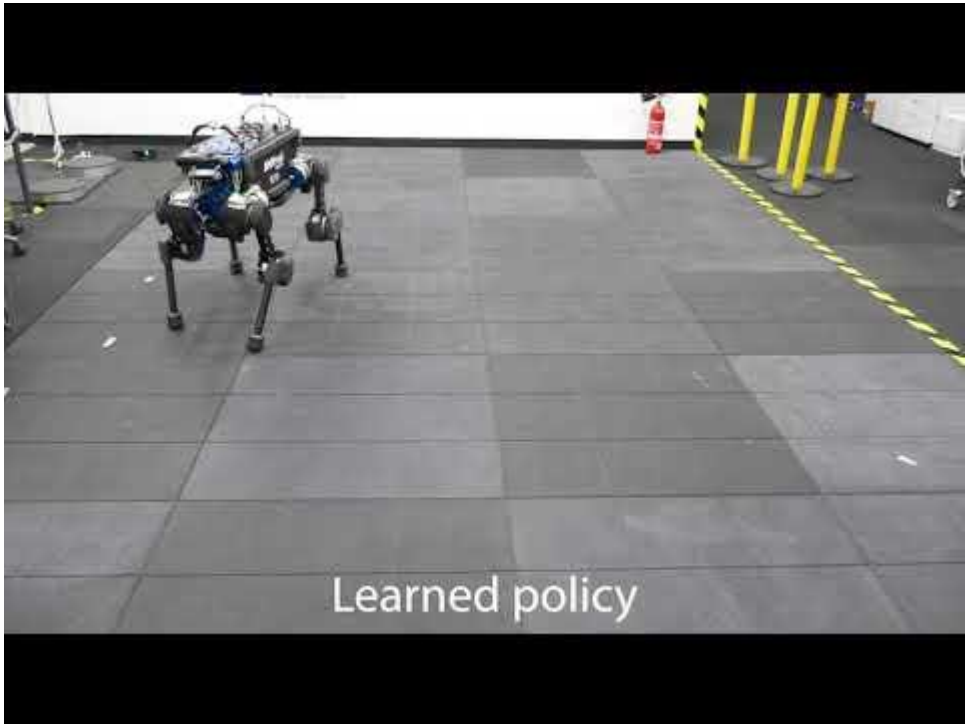
Exp 1: Command-conditioned locomotion

- Commands - forward velocity, lateral velocity, and yaw rate - Angular Velocity.
- Reward function
  - Angular velocity , moving speed, torque, joint speed
- Learning time - 4 hours in the real world

- Results - Compared the obtained results with a models based approach
  - Learned Policy -> Less torque & mechanical power

Exp 1: Command-conditioned locomotion





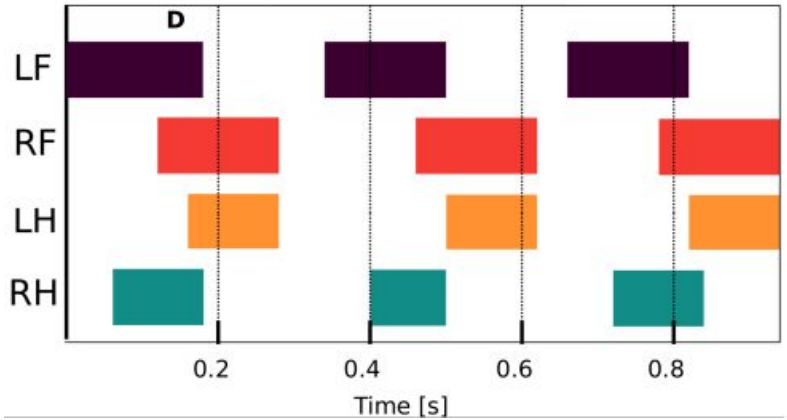
Exp 2: High speed Locomotion

- Task is to run as fast as possible
- Reward function
  - Angular velocity , moving speed, torque, joint speed
- Learning time - 4 hours in the real world

- Results
  - Achieved maximum speed compared to the previous research
  - Different gait pattern
  - Exploited full hardware capacity to achieve the goal

Exp 2: High speed Locomotion

- Flight duration is more



Exp 3: Recovery from a fall

- Task is to get up from a random configuration
- Reward function
  - Join acceleration, constraints on torque, joint speed ..
- Learning Time – 11 hours in the real world

Exp 3: Recovery from a fall

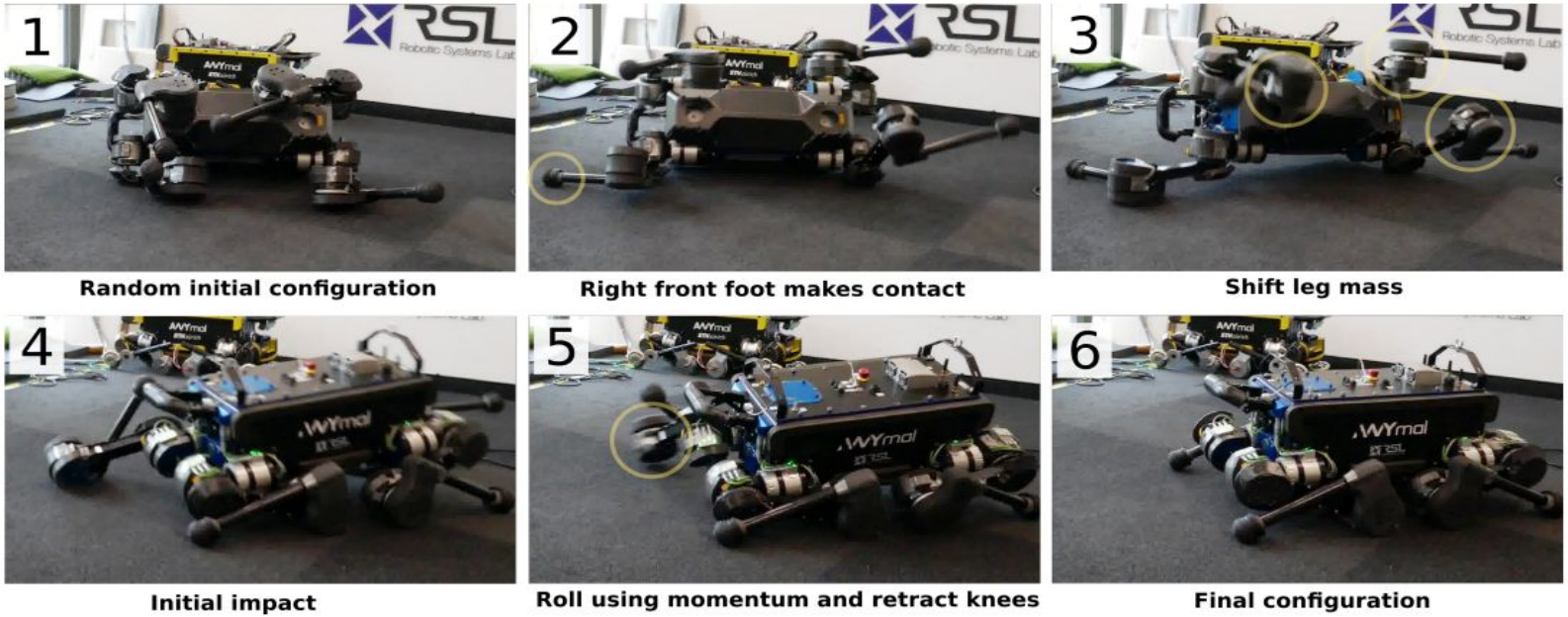


Fig. 4. A learned recovery controller deployed on the real robot. The learned policy successfully recovers from a random initial configuration in less than 3 seconds.

Exp 3: Recovery from a fall

Discussion of Results

- Same policy used for 3 months - policy is robust to change in hardware because of wear and tear
- Learned actuator dynamics significantly reduced the reality gap
- Less computation time - (25 micro sec on single CPU thread)

Critique / Limitations / Open Issues

- Computation required for the both training and inference is not so large and can be controlled.
  - Utilized 1 CPU and 1 GPU

- Limitations and open issues -
  - Requires human expertise for tuning and to design initial state distribution for each new task.
  - Possibility of overfitting while training the actuator net
  - Policies are not generalized to multiple tasks

Future Work for Paper

- Technique to improve the reward design and decide on the initial state distribution would be helpful
- Can try to perform multiple tasks by giving hierarchical structure control policy

Extended Readings

- Learning to Walk via Deep Reinforcement Learning
- A Survey on Policy Search Algorithms for Learning Robot Controllers in a Handful of Trials
- Solving Rubik's Cube with a Robot Hand

Summary

- Problem -> Sim to real transfer
- Importance -> Optimal time, Less risky training
- Difficult -> Creating simulation close to reality is hard.
- Limitation of prior work -> Designing the controllers is a laborious task, the models are inaccurate for complex robots
- Key insight of the proposed work -> Modelling actuator with NN; Making the policy robust.
- What did they demonstrate by this insight? Outperformed existing model based controllers

References

- Learning agile and dynamic motor skills for legged robots
- Notes on the paper by Sharath Chandra
- Referred presentation

Initial state - Exp1 & Exp2

Sampled from previous trajectory or from the table

	mean	standard deviation
base position	$[0, 0, 0.55]^T$	1.5 cm
base orientation	$[1, 0, 0, 0]^T$	0.06 rad (about a random axis)
joint position	$[0, 0.4, -0.8, 0, 0.4, -0.8, 0, -0.4, 0.8, 0, -0.4, 0.8]^T$	0.25 rad
base linear velocity	$\mathbf{0}^3$	0.012 m/s
base angular velocity	$\mathbf{0}^3$	0.4 rad/s
joint velocity	$\mathbf{0}^{12}$	2 rad/s

Table S3. Initial state distribution for training the command-conditioned and high-speed locomotion controllers. The initial state is randomized to make the trained policy more robust.

Initial state - Exp3

Sampled from previous trajectory or from the table

we dropped ANYmal from a height of 1.0 m with randomized orientations and joint positions, ran the simulation for 1.2 s, and used the resulting state as initialization

joint velocities measurement is inaccurate in the real robot so they add noise to the joint velocies in simulation for robustness

ANYmal details

ANYmal is equipped with 12 SEAs [60, 61]. An SEA is composed of an electric motor, a high gear ratio transmission, an elastic element, and two rotary encoders to measure spring deflection and output position