

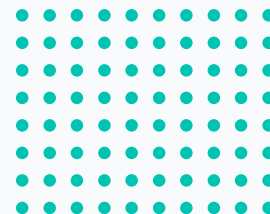


Identifying Conditions for Autonomous Dynamic Gait Transitions in Quadruped Robots

Intelligent Robotics

Emanuele Cuzzocrea

Prof. Alberto Finzi



Index

01

Introduction

02

Reward Machines

03

RM for Quadruped
Locomotion

04

Robust gait
transitions

05

Personal
contribution

06

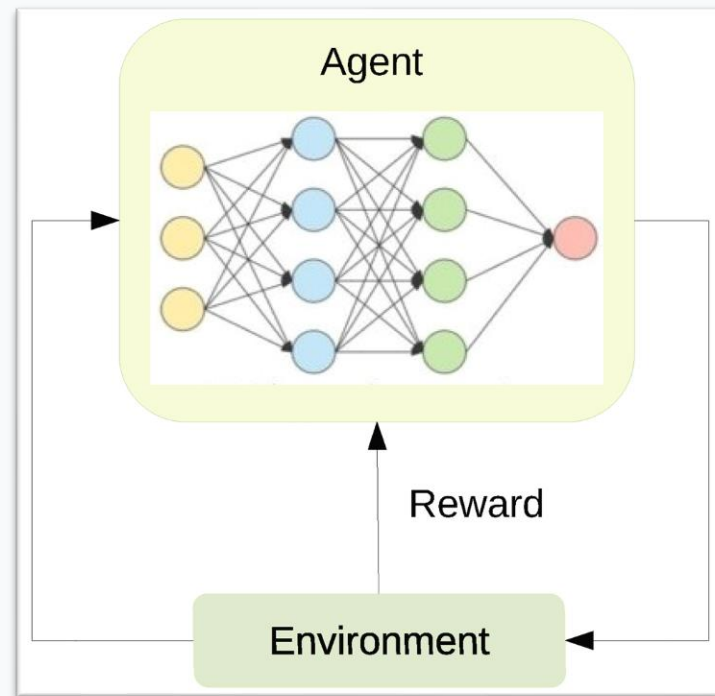
Conclusions





1. Introduction

- Using deep reinforcement learning for controlling a quadruped robot
- Use reward machines for learning different gaits
- Explore transitions of policies
- Condition for autonomous transitions for the higher level control system

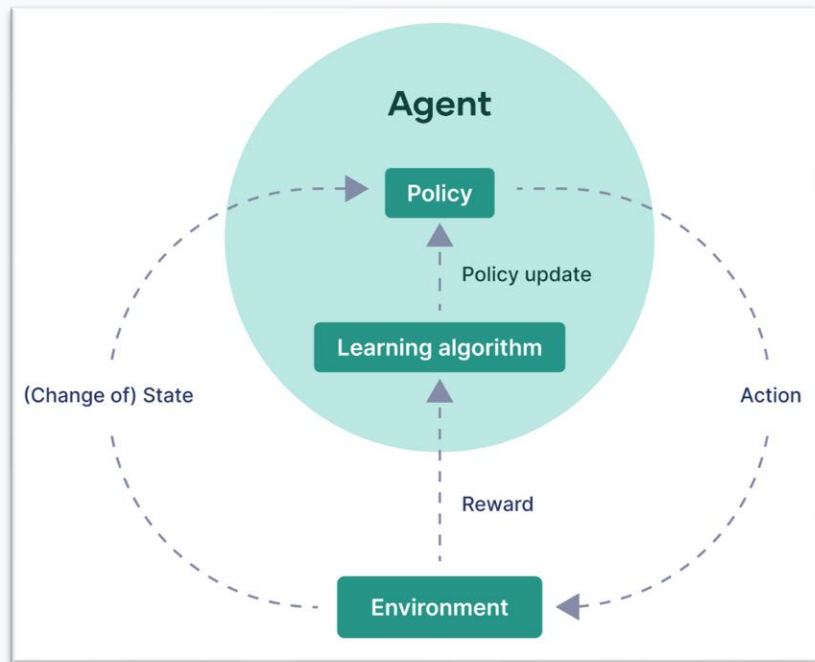




Reinforcement learning

$$M = (S, A, T, R, \gamma)$$

$$\pi : S \rightarrow A$$





Robot used: ANYmal B

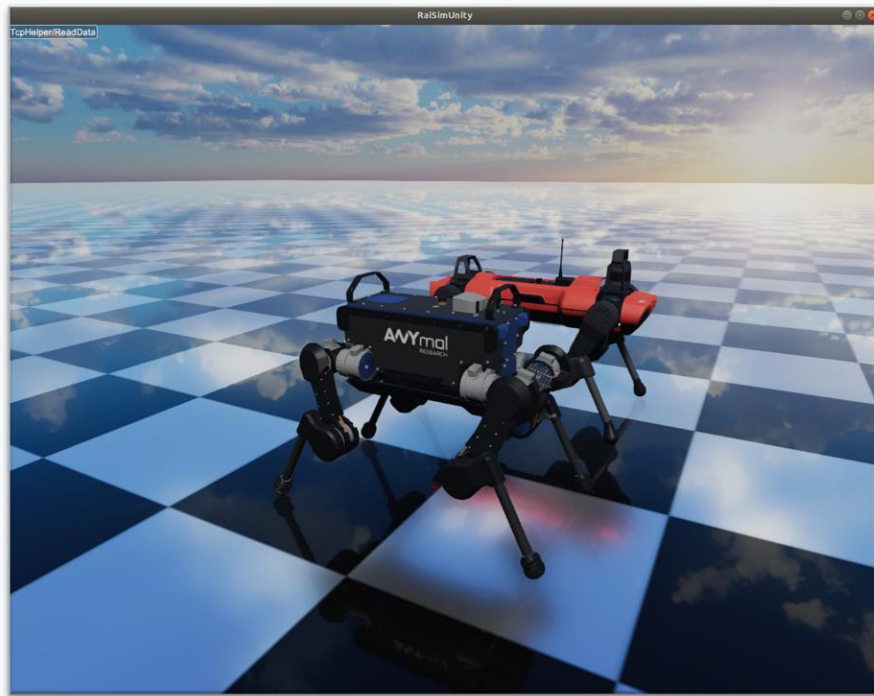
- Quadruped robot developed by ANYbotics
- 4 legs with each three identical motors (ANYdrives): electrically actuated and can sense force
- Compliant walking
- Depth cameras, LIDAR, ...





Simulator: Raisim

- Cross-Platform multi-body physics engine for robotics and AI
- Deep Reinforcement Learning
- Uses Unity or Unreal for visualization
- Pytorch
- Best, after Isaac Gym from NVIDIA





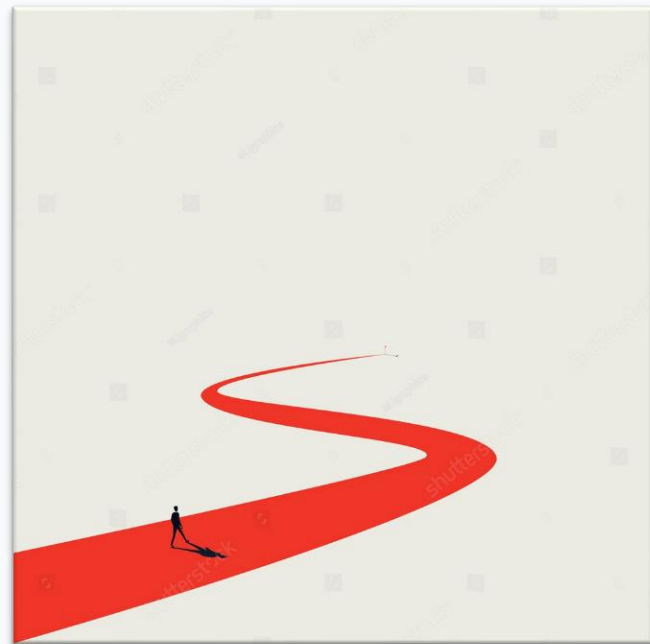
Necessity of different gaits





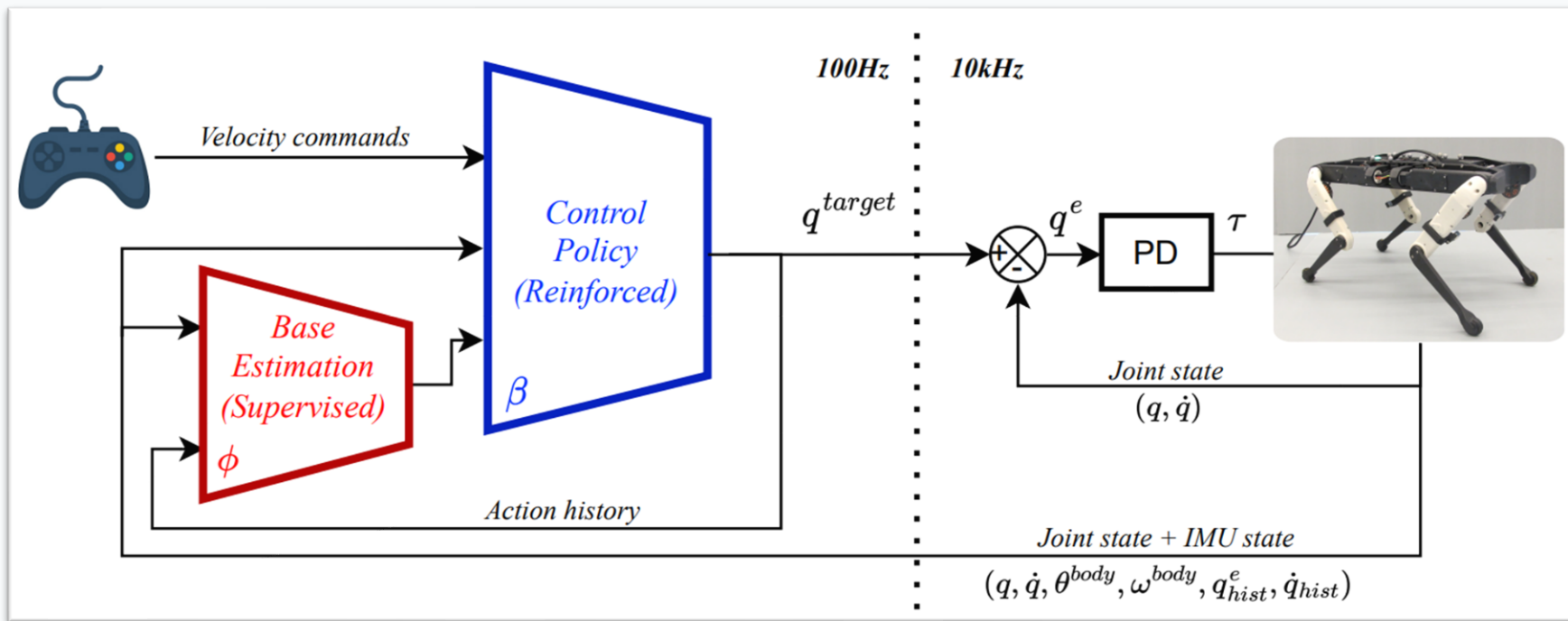
2. Reward Machines

- Used to manage milestone sub-goals in larger tasks
- Addressing issues of sparse or non-Markovian reward functions
- Specify sub-goals through a finite-state automaton
- Keeps the reward function Markovian
- Can speed up the training phase





3. RM for Quadruped Locomotion





State space

- $q \rightarrow$ 12 joint angles
- $\dot{q} \rightarrow$ 12 joint velocities
- $z \rightarrow$ Body height
- $\phi \rightarrow$ Body orientation
- $v \rightarrow$ Body linear velocity
- $\omega \rightarrow$ Body angular velocity
- $u \rightarrow$ Current RM state
- $\delta \rightarrow$ Number of steps since previous RM state change
- $P \rightarrow$ Vector of four boolean variables, that are 1 if the corresponding foot is touching the ground, 0 otherwise

$$S = (q, \dot{q}, z, \phi, v, \omega, u, \delta, P)$$



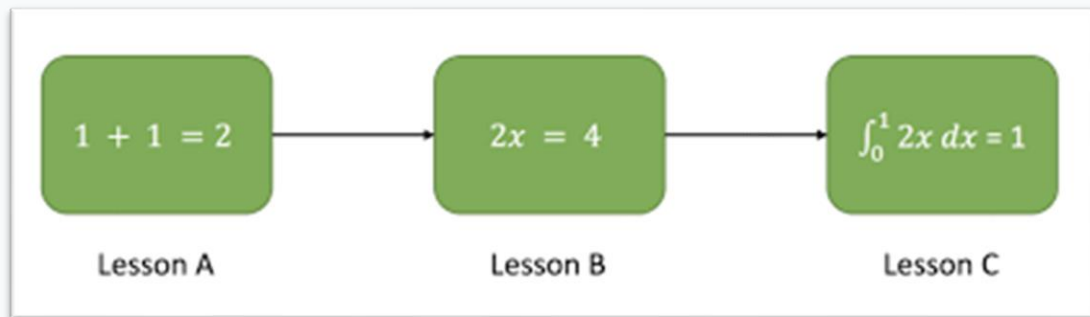
Reward function (fixed forward velocity)

Term Description	Definition	Wheight
Linear Velocity x	$\exp(- v_{d,x} - v_x ^2)$	10
Linear Velocity y	v_y^2	-10
Angular Velocity x, y	$ \omega_{x,y} ^2$	-0.5
Angular Velocity z	$ \omega_z ^2$	-25
Joint Torques	$ \tau ^2$	-0.00004
Joint Position	$ q - q_{init} ^2$	-0.1
Joint Velocity	$ \dot{q} ^2$	-0.01



Curriculum learning

- Increase slowly the weight of the penalties
- The robot can find a local reward maximum by doing nothing!
- Fundamental when not using reward machines
- Optional when using reward machines
- The training can become very slow





Training of trot gait

Time: 1.27999

Iteration: 0



anyamal

Joint State

- ☐ Display Joints
- ☐ Display COM

ROOTp_x	pos:0.0608
(float)gv:9	vel:0.0026
ROOTp_y	pos:0.0650
(float)gv:10	vel:0.4544
ROOTp_z	pos:0.4517
(float)gv:11	vel:-0.2262
ROOTa_x	pos:-0.1126
(float)gv:6 gc:7	vel:0.7542
ROOTa_y	pos:-0.0588
(float)gv:7 gc:8	vel:0.8403
ROOTa_z	pos:-0.0293
(float)gv:8 gc:9	vel:1.0841
LF_HAA	pos:-0.2829
(rev)gv:6 gc:7	vel:-3.2027
LF_HFE	pos:0.4268
(rev)gv:7 gc:8	vel:-0.2100
LF_KFE	pos:-1.0149
(rev)gv:8 gc:9	vel:-1.8849
RF_HAA	pos:0.2319
(rev)gv:9 gc:10	vel:-2.5987
RF_HFE	pos:0.3485
(rev)gv:10 gc:11	vel:1.9361
RF_KFE	pos:-0.7862
(rev)gv:11 gc:12	vel:-4.0426
LH_HAA	pos:-0.2594
(rev)gv:12 gc:13	vel:-0.2849
LH_HFE	pos:-0.4035
(rev)gv:13 gc:14	vel:-1.1338
LH_KFE	pos:1.2603
(rev)gv:14 gc:15	vel:-0.4340
RH_HAA	pos:0.2157



Training of bound gait

Time: 1.62999

Iteration: 0



anymal

Joint State

- ☐ Display Joints
- ☐ Display COM

ROOTp_x	pos:-0.0170
(float)gv:9	vel:-0.0080
ROOTp_y	pos:0.0064
(float)gv:10	vel:0.0137
ROOTp_z	pos:0.4687
(float)gv:11	vel:-0.5985
ROOTa_x	pos:-0.0294
(float)gv:6 gc:7	vel:-1.0100
ROOTa_y	pos:-0.2501
(float)gv:7 gc:8	vel:-5.4339
ROOTa_z	pos:-0.0110
(float)gv:8 gc:9	vel:0.1540
LF_HAA	pos:-0.1716
(rev)gv:6 gc:7	vel:3.3922
LF_HFE	pos:0.4108
(rev)gv:7 gc:8	vel:-0.1251
LF_KFE	pos:-0.4848
(rev)gv:8 gc:9	vel:10.6199
RF_HAA	pos:0.2495
(rev)gv:9 gc:10	vel:1.7372
RF_HFE	pos:0.3614
(rev)gv:10 gc:11	vel:3.1272
RF_KFE	pos:-0.2960
(rev)gv:11 gc:12	vel:10.3476
LH_HAA	pos:-0.4296
(rev)gv:12 gc:13	vel:0.5609
LH_HFE	pos:-0.9045
(rev)gv:13 gc:14	vel:-3.2130
LH_KFE	pos:1.9202
(rev)gv:14 gc:15	vel:4.4427
RH_HAA	pos:0.1543



Training of pace gait

Time: 1.16999

Iteration: 0



anyamal

Joint State

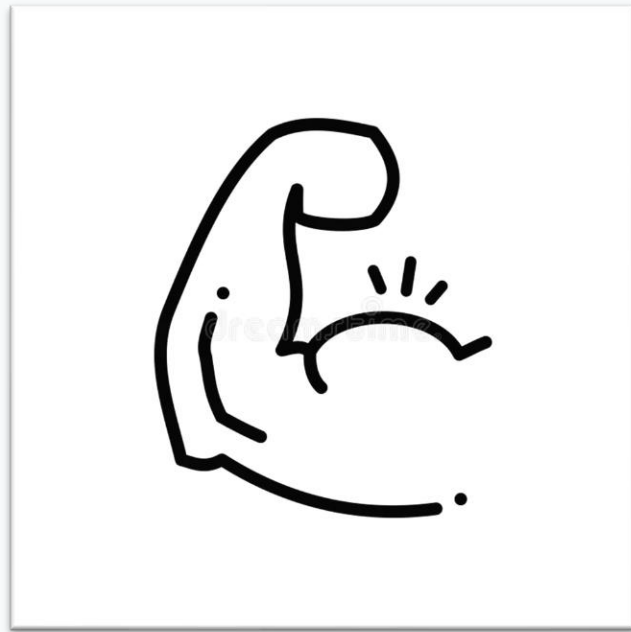
- ☐ Display Joints
- ☐ Display COM

ROOTp_x (float)gv:9	pos:-0.0085
ROOTp_y (float)gv:10	vel:-0.0322
ROOTp_z (float)gv:11	pos:0.0525
ROOTa_x (float)gv:6 gc:7	vel:0.3315
ROOTa_y (float)gv:7 gc:8	pos:0.4793
ROOTa_z (float)gv:8 gc:9	vel:-0.0413
LF_HAA (rev)gv:6 gc:7	pos:-0.0876
LF_HFE (rev)gv:7 gc:8	vel:0.2073
LF_KFE (rev)gv:8 gc:9	pos:-0.0601
RF_HAA (rev)gv:9 gc:10	vel:0.8127
RF_HFE (rev)gv:10 gc:11	pos:0.0799
RF_KFE (rev)gv:11 gc:12	vel:-0.8014
LH_HAA (rev)gv:12 gc:13	pos:-0.3431
LH_HFE (rev)gv:13 gc:14	vel:-0.5153
LH_KFE (rev)gv:14 gc:15	pos:0.4995
RH_HAA	vel:0.3119



4. Robust gait transitions

- Specific training can be done on transitions
- Domain randomization
- External disturbances





Domain and dynamic randomization

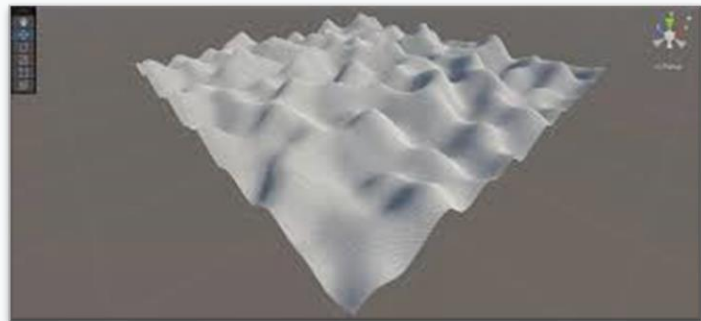
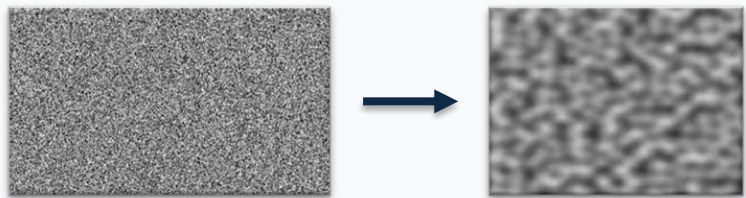
- Significantly improve the sim-to-real gap
- Make the locomotion more robust
- Can be implemented with curriculum learning
- Adding noise to the state observation
- Adding noise to the motors
- Adding noise to the parameters of control (K_p, K_d)
- Randomize the terrain
- Randomize the obstacles
- Randomize the dynamics





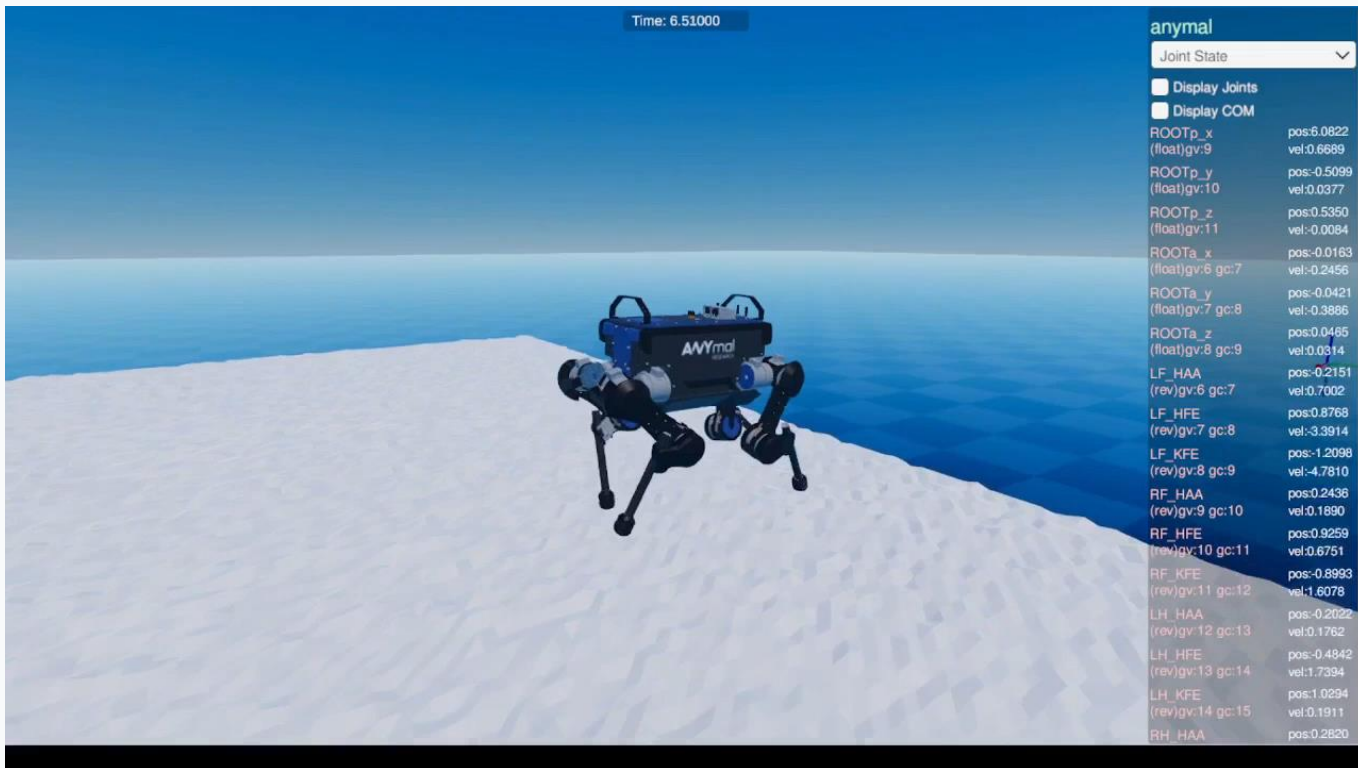
Terrain randomization with Perlin noise

- Type of gradient noise developed by Ken Perlin in 1983
- Many uses: procedurally generating terrain, pseudo-random changes to variables, assisting in the creation of image textures
- Can be defined for any number of dimensions
- Change terrain each 5 iterations for limiting computational cost





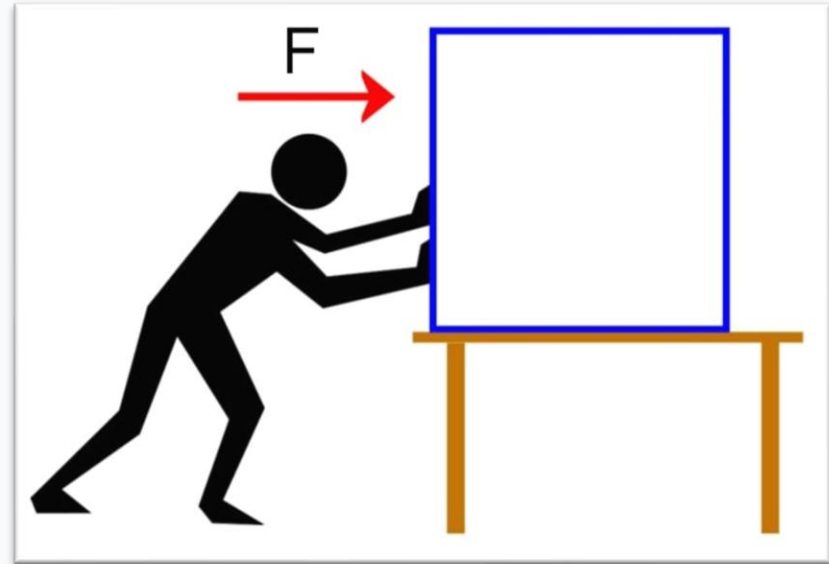
Terrain randomization with Perlin noise





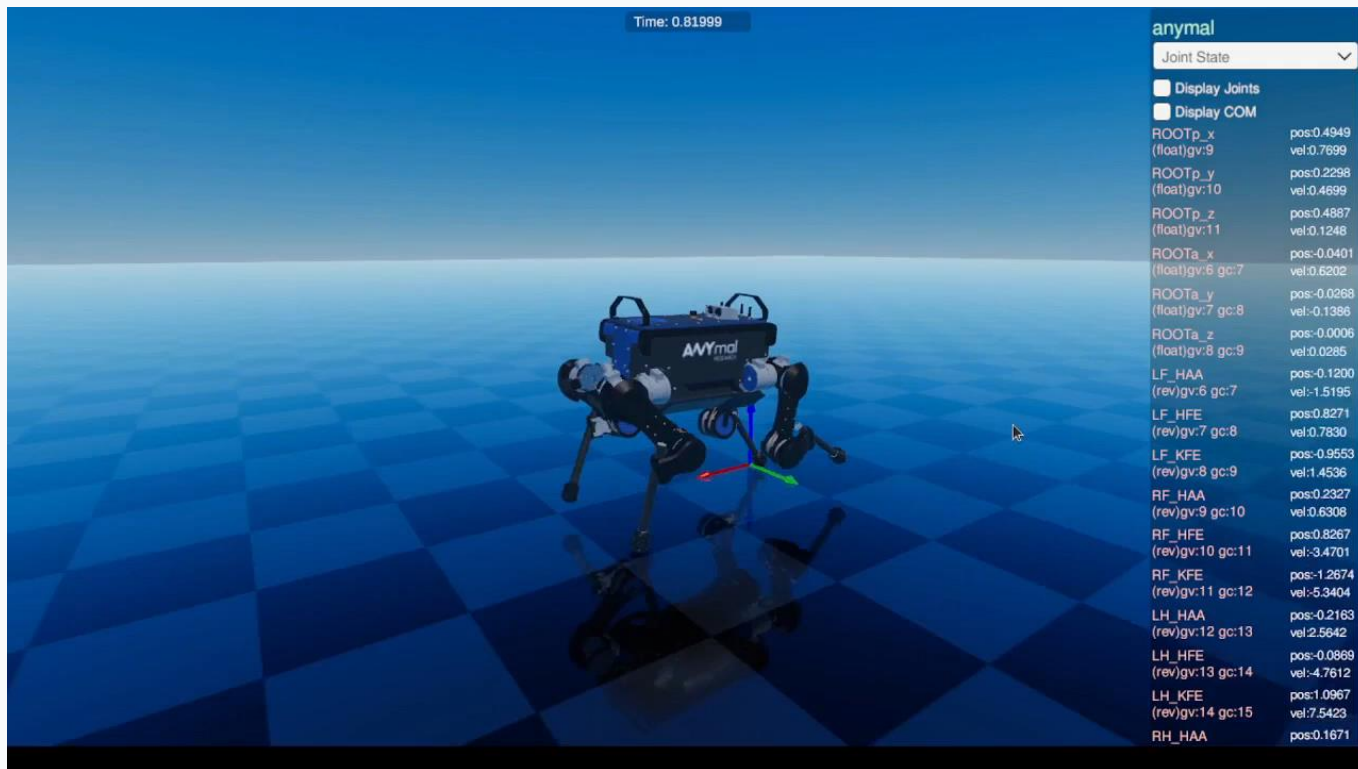
External disturbances

- Random forces along x, y, z
- Used to recover from difficult position
- Applied to the base frame
- Applied on the legs
- Preparation for the real world



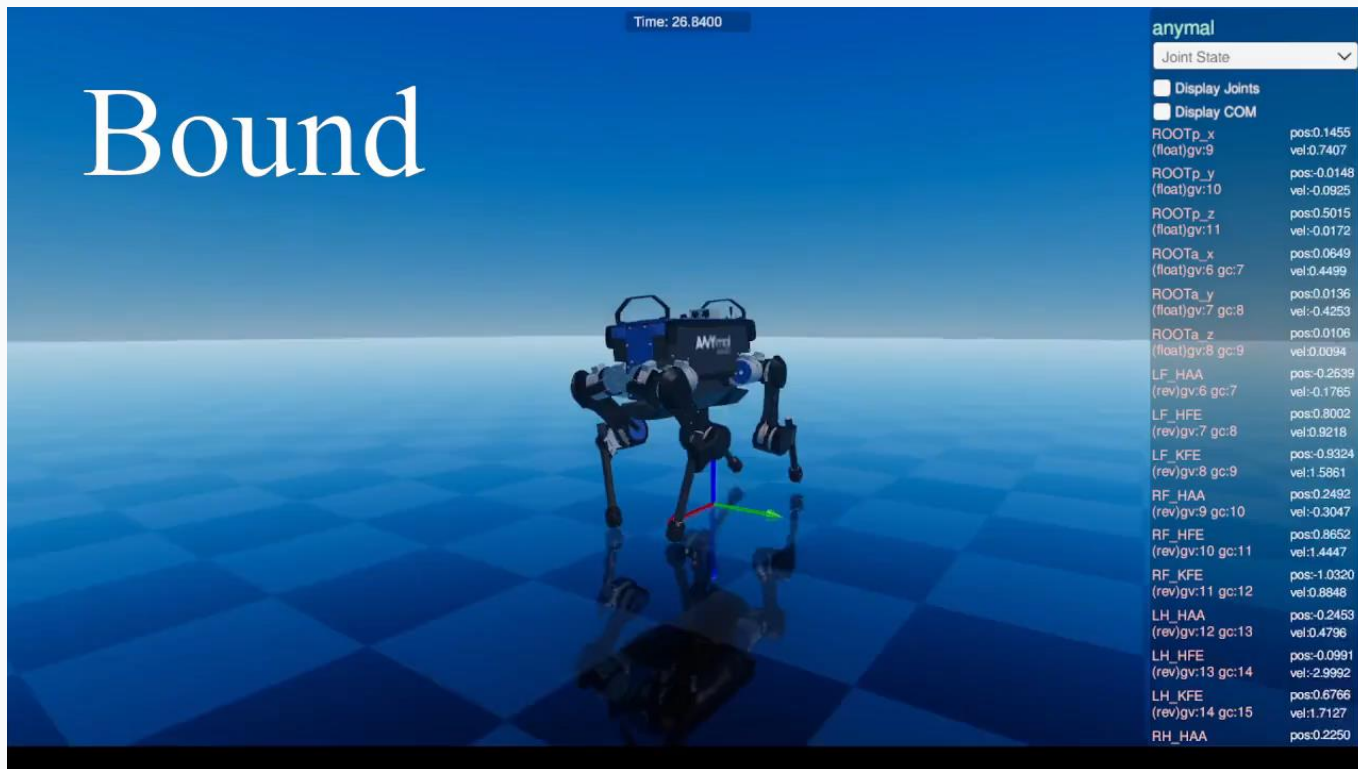


External disturbances (trot gait example)





Gait transitions example (manual switching)





5. Personal contribution: Condition for autonomous transition

- We need a simple condition to be checked so that the **higher-level controller** knows when to perform the policy change, all autonomously.
- Let's take in a vector $h = \{h_{FL}, h_{FR}, h_{HL}, h_{HR}\}$ the height of each foot last time they touched the ground.
- Only two parameters to tune:
 - $\alpha \rightarrow$ Magnitude of the irregularity of the terrain
 - $\beta \rightarrow$ Length of the irregularity of the terrain
 - $k \rightarrow$ Just a counting variable



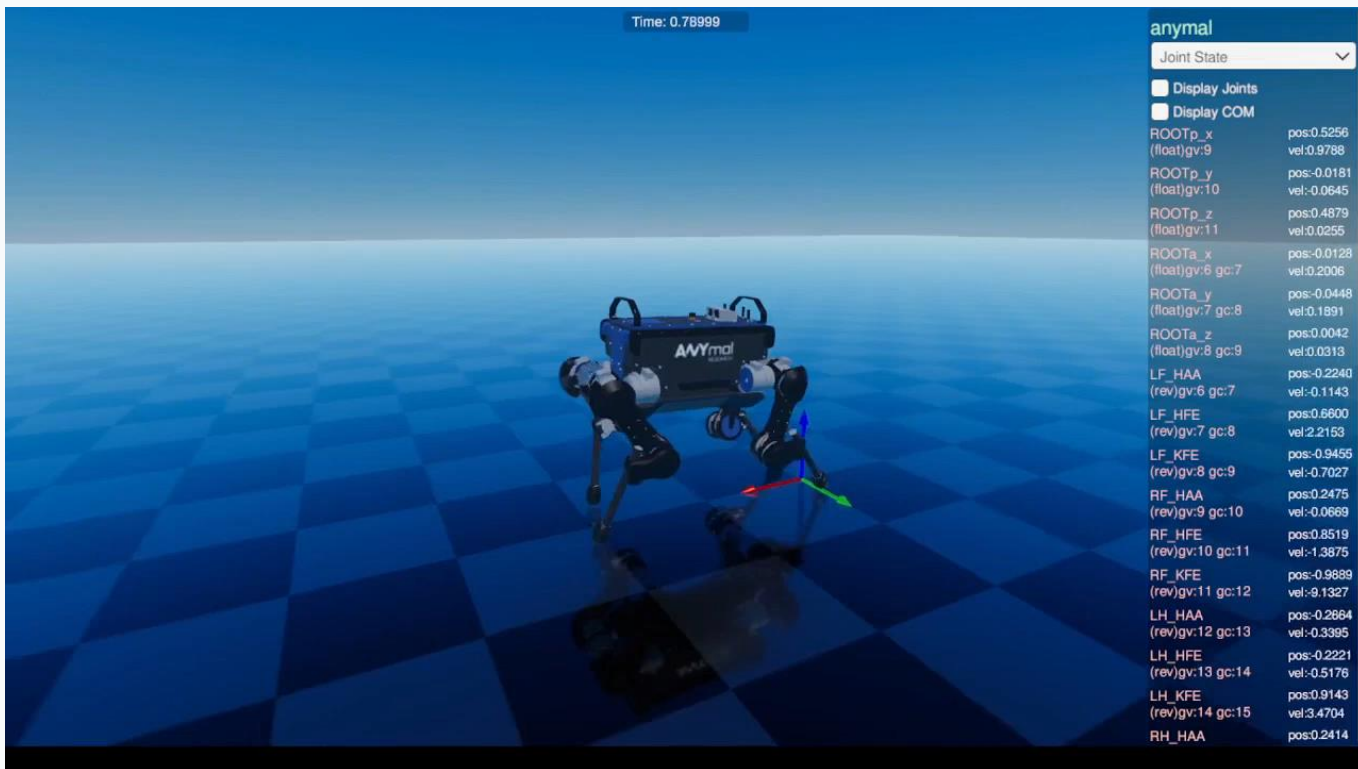
Pseudocode

Algorithm 1 Condition for autonomous gait transitions

```
Run at each step:  
for all  $i, j$  do  
    if  $|h_i - h_j| > \alpha$  then  
         $k = k + 1$   
    end if  
end for  
if  $k > \beta$  then  
    Change policy  
end if
```

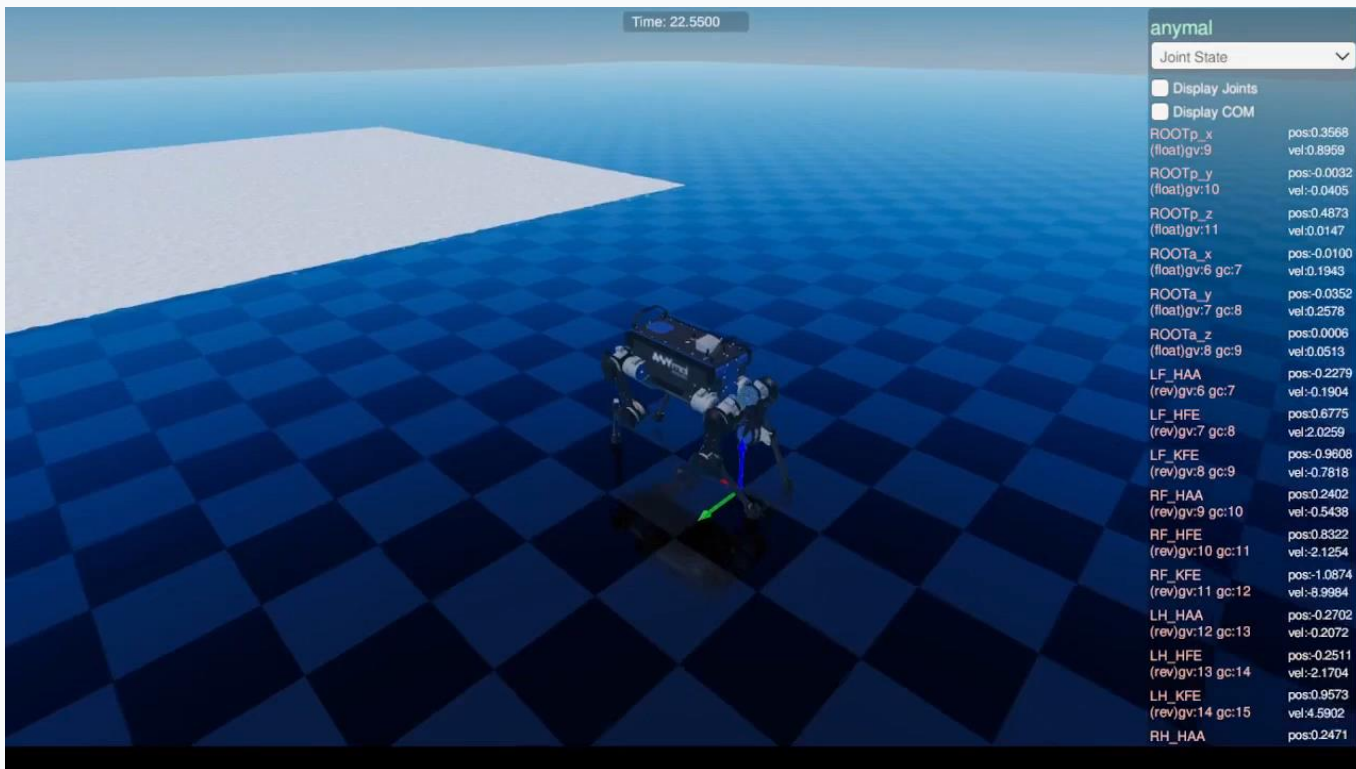


Autonomous change of gait ($\alpha = 0.2, \beta = 1000$)





Autonomous change of gait (aerial view)





6. Conclusions

- Single gaits were successfully learned thanks to deep reinforcement learning techniques
- Reward Machines turned out to be crucial for the right gait learning
- Transition between gaits were made robust randomizing the terrain, and applying external forces to the robot
- Condition for autonomous gait change proved to be suitable for different terrain, and easy to tune according to the objective



Thanks.

