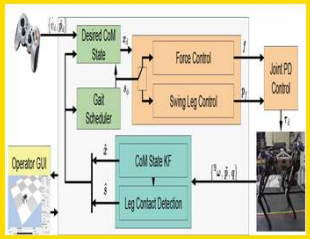
DIPARTIMENTO DI INGEGNERIA ELETTRICA E TECNOLOGIE DELL'INFORMAZIONE
SCUOLA POLITECNICA E DELLE SCIENZE DI BASE

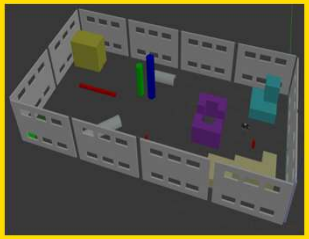
FIELD AND SERVICE ROBOTICS – Technical Report

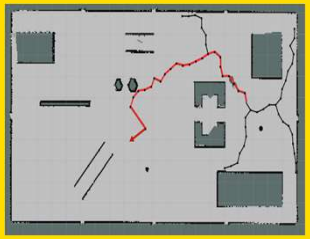
Model-Based and Data-Driven Control of a Quadruped Robot

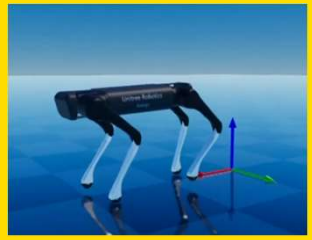
Emanuele Cuzzocrea, Vito Daniele Perfetta

Prof. Fabio Ruggiero









INTRODUCTION

- Quadruped Robots → A1 and Aliengo
- Model-based Control → MPC and Gait Analysis
- Path Planning → Rapidly Exploring Random Tree (RRT)
- Data-driven Control → Reward Machines and Gait Analysis
- Conclusions

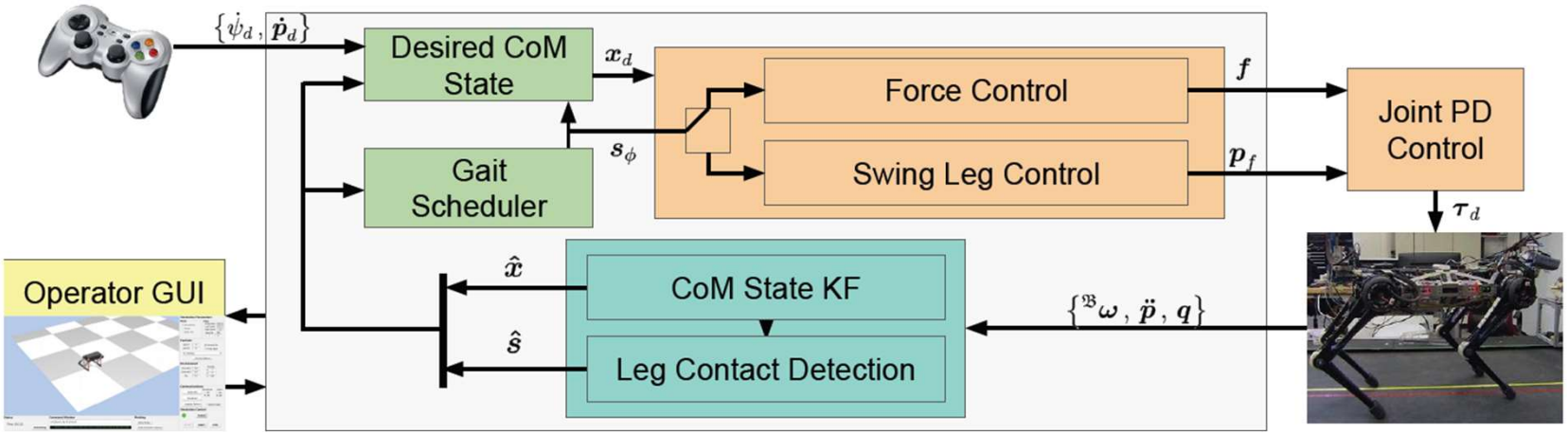
QUADRUPED ROBOTS

Unitree A1 Dog Robot

Unitree Aliengo Dog Robot

- Used for the Model-based control
- Used for the Data-driven control

MODEL-BASED CONTROL



- The operator provides desired translational velocity and turning rate
- These commands are used to plan the CoM reference trajectory
- 2 controllers: swing leg and stance leg control

MODEL-BASED CONTROL

- Swing Leg Control

$$\tau_i = J_i^T \left[K_p (p_{i.ref}^b - p_i^b) + K_d (v_{i,ref}^b - v_i^b) \right] + \tau_{i,ff}$$

$$\tau_{i,ff} = J_i^T \Lambda_i \left(a_{i,ref}^b - \dot{J}_i \dot{q}_i \right) + C_i \dot{q}_i + G_i$$

- Stance Leg Control

$$\tau_i = J_i^T R_i f_i$$

$$f_i = MPC(x)$$

MODEL-BASED CONTROL

- Neglecting the leg dynamics, the simplified robot dynamic model is

$$\frac{d}{dt}\begin{bmatrix}\hat{\Theta}\\ \hat{p}\\ \hat{\omega}\\ \hat{\dot{p}}\end{bmatrix} = \begin{bmatrix}0_3 & 0_3 & R_z(\psi) & 0_3\\ 0_3 & 0_3 & 0_3 & 1_3\\ 0_3 & 0_3 & 0_3 & 0_3\\ 0_3 & 0_3 & 0_3 & 0_3\end{bmatrix}\begin{bmatrix}\hat{\Theta}\\ \hat{p}\\ \hat{\omega}\\ \hat{\dot{p}}\end{bmatrix} + \begin{bmatrix}0_3 & \dots & 0_3\\ 0_3 & \dots & 0_3\\ \hat{I}^{-1}[r_1]_\times & \dots & \hat{I}^{-1}[r_n]_\times\\ 1_3/m & \dots & 1_3/m\end{bmatrix}\begin{bmatrix}f_1\\ \dots\\ \dots\\ f_n\end{bmatrix} + \begin{bmatrix}0\\ 0\\ 0\\ g\end{bmatrix}$$

- Adding the gravity term in the state vector, i.e. $x(t) = [\hat{\Theta} \quad \hat{p} \quad \hat{\omega} \quad \hat{\dot{p}} \quad g]^T$, $u_i(t) = f_i$

$$\dot{x}(t) = A_c(\psi)x(t) + B_c(r_1, \dots, r_n, \psi)u(t)$$

- Subject to the following constraints:
  - null swing feet ground reaction force
  - limits on the minimum and maximum GRF_z
  - pyramid approximation of the friction cone

$$f_{min} \leq f_z \leq f_{max}$$

$$-\mu f_z \leq \pm f_x \leq -\mu f_z$$

$$-\mu f_z \leq \pm f_y \leq -\mu f_z$$

MODEL PREDICTIVE CONTROL

$$\min_{x,u}\left\{\sum_{i=0}^{k-1}\left(\left\|x_{i+1}-x_{i+1,ref}\right\|_Q+\|u_i\|_R\right)\right\}$$

$$s.t.$$

$$x_{i+1}=A_i x_i+B_i u_i \quad \forall i=0,\dots,k-1$$

$$\underline{c}_i \le C_i u_i \le \overline{c}_i \quad \forall i=0,\dots,k-1$$

$$D_i u_i = 0 \quad \forall i=0,\dots,k-1$$

$$x_0 = x(t)$$

$$\min_U\left\{\frac{1}{2}U^T H U+U^T g\right\}$$

$$s.t.$$

$$\underline{c} \le CU \le \overline{c}$$

$$where$$

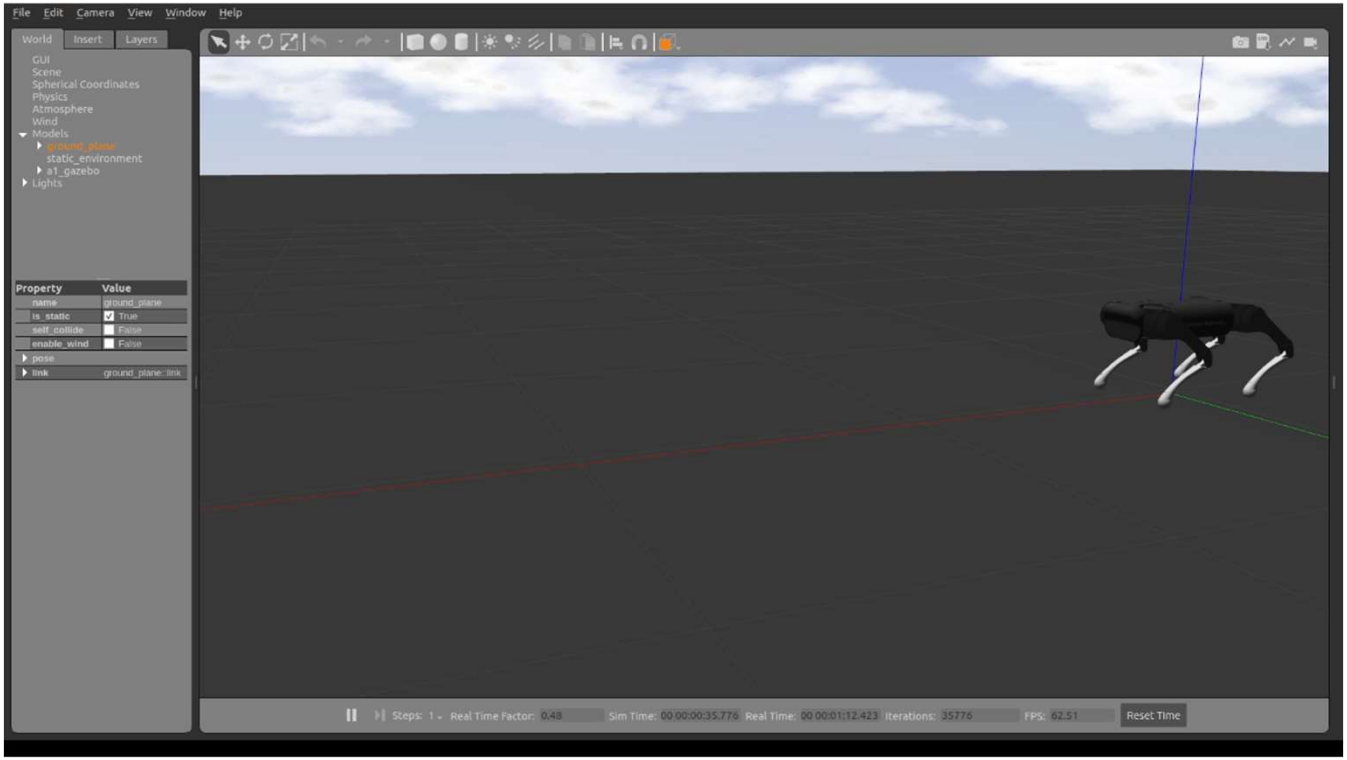
$$H = 2(B_{qp}^T L B_{qp}+K)$$

$$g = 2B_{qp}^T L(A_{qp}x_0 - x_{ref})$$

$$L = diag_k(Q)$$

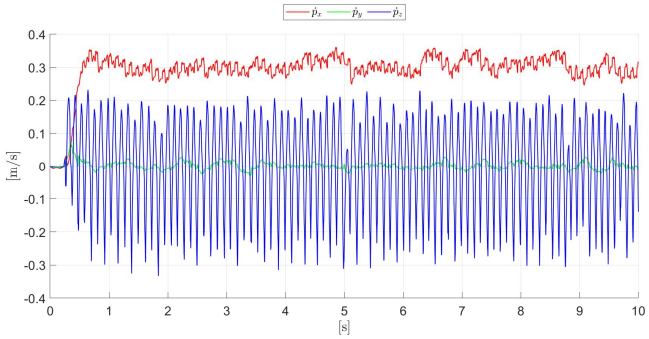
$$K = diag_k(R)$$

GAIT ANALYSIS – TROT

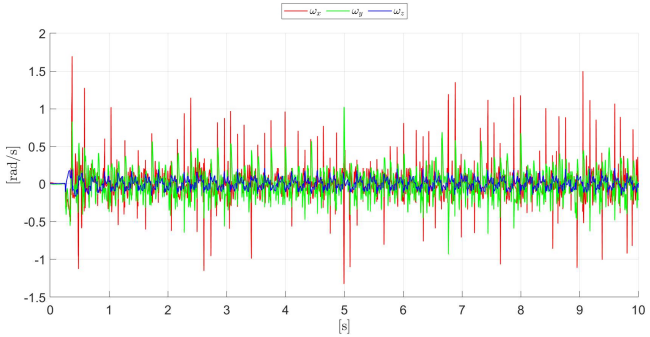


$$v_{x,d} = 0.3 \text{ m/s}$$

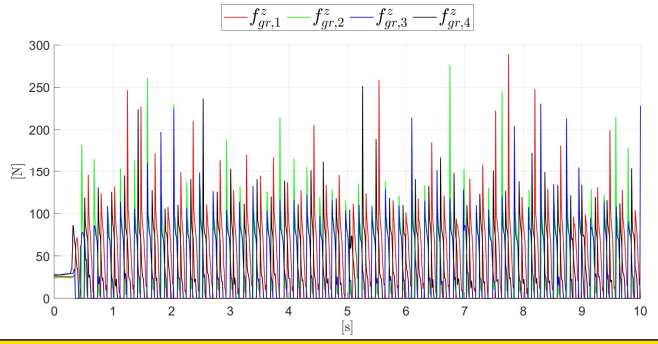
Linear Velocities



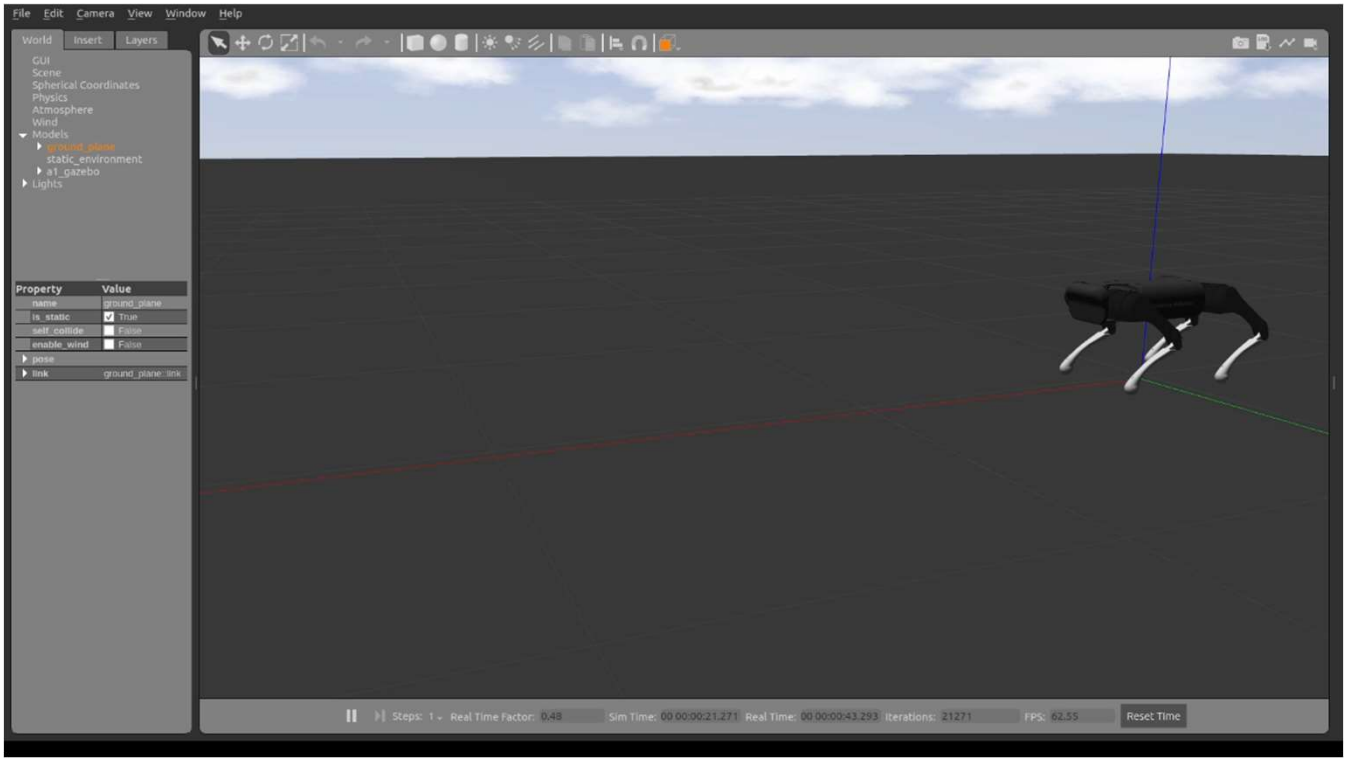
Angular Velocities



Ground Reaction Forces

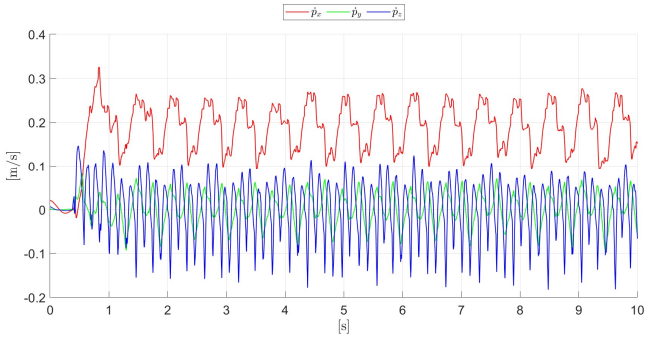


GAIT ANALYSIS – CRAWL

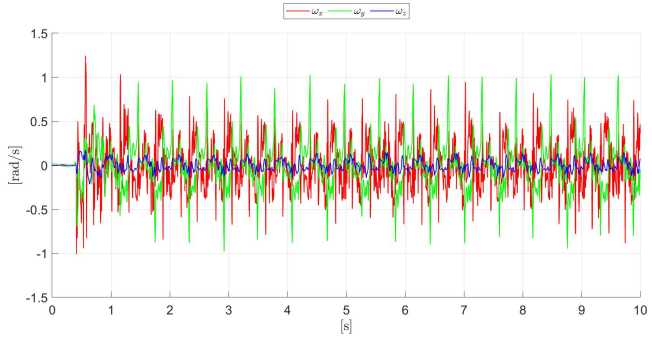


$$v_{x,d} = 0.2 \text{ m/s}$$

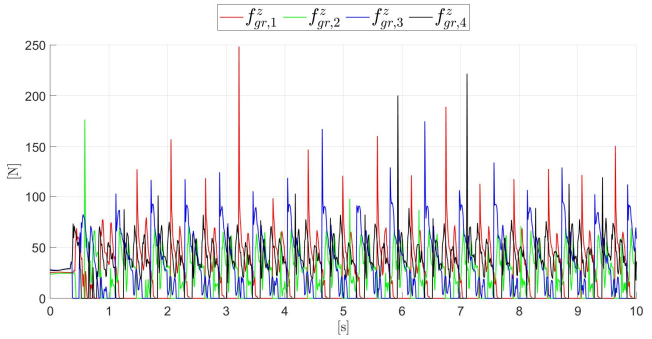
Linear Velocities



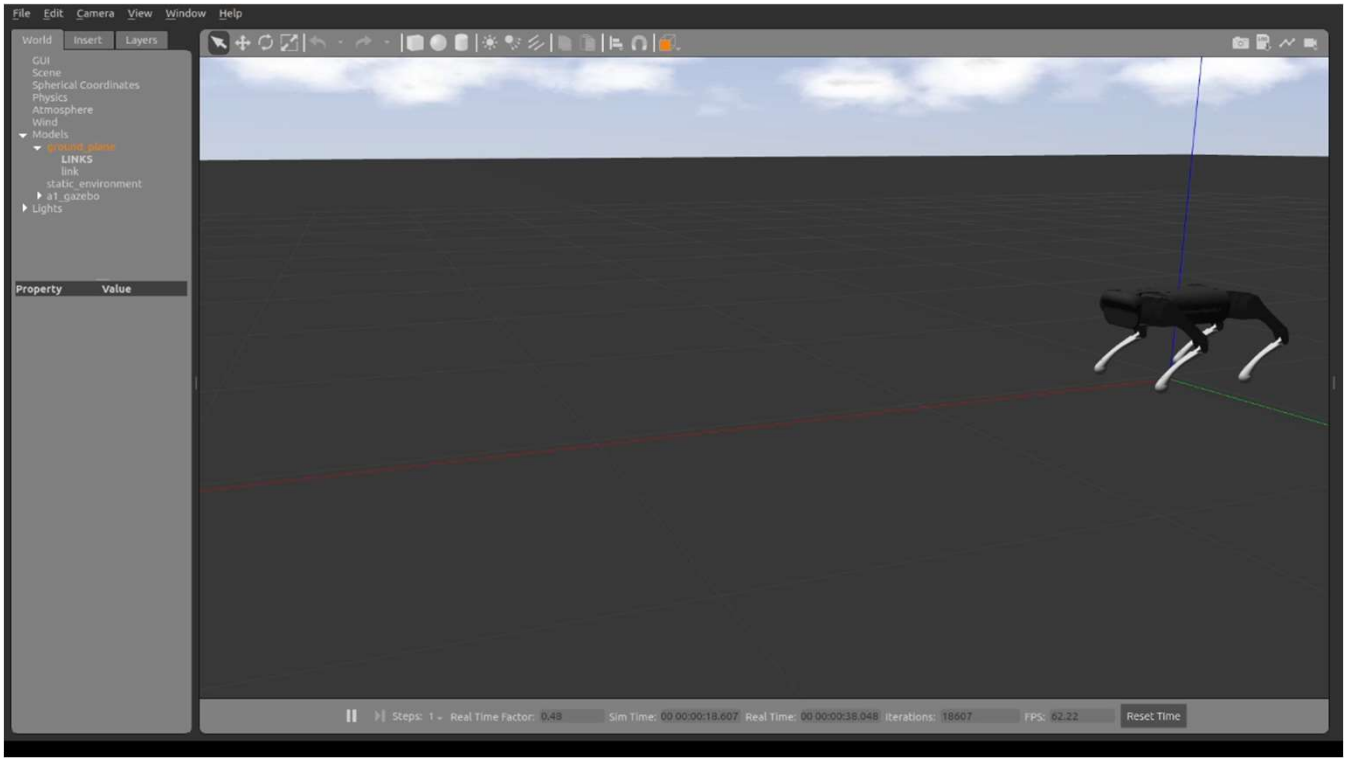
Angular Velocities



Ground Reaction Forces

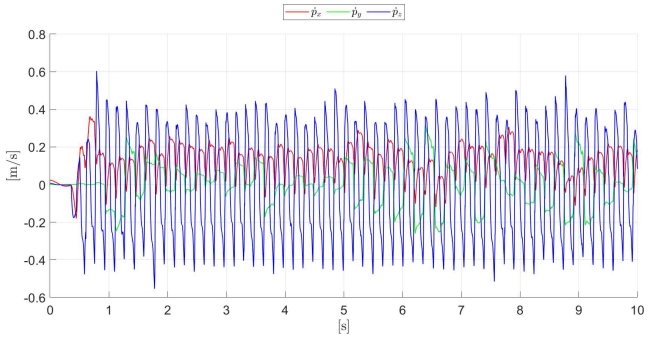


GAIT ANALYSIS – BOUND

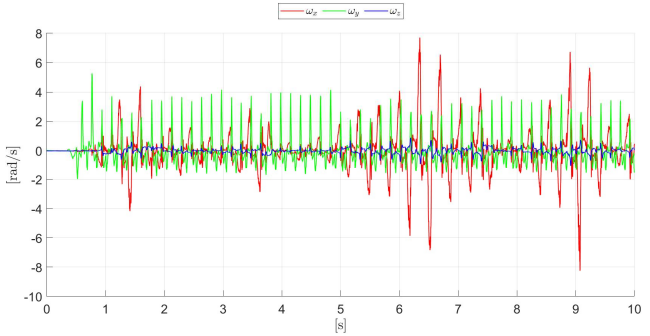


$$v_{x,d} = 0.3 \text{ m/s}$$

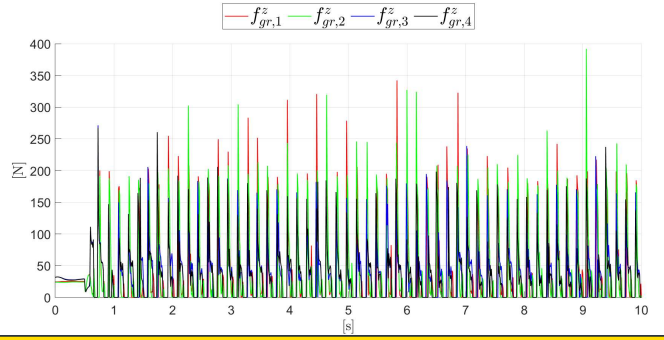
Linear Velocities



Angular Velocities



Ground Reaction Forces



ADDING NEW GAITS

- Gait implementation:
  - stance_duration: amount of stance time for each leg in a gait cycle.
  - duty_factor: fraction of stance phase in the gait cycle.
  - init_phase_full_cycle: initial phase of each leg at the beginning of the full cycle.
  - initial_leg_state: initial state of each leg, where 1 indicates "STANCE" and 0 indicates "SWING".
  - contact_detection_phase_threshold: contact threshold when the leg state switches from "SWING" to "STANCE".

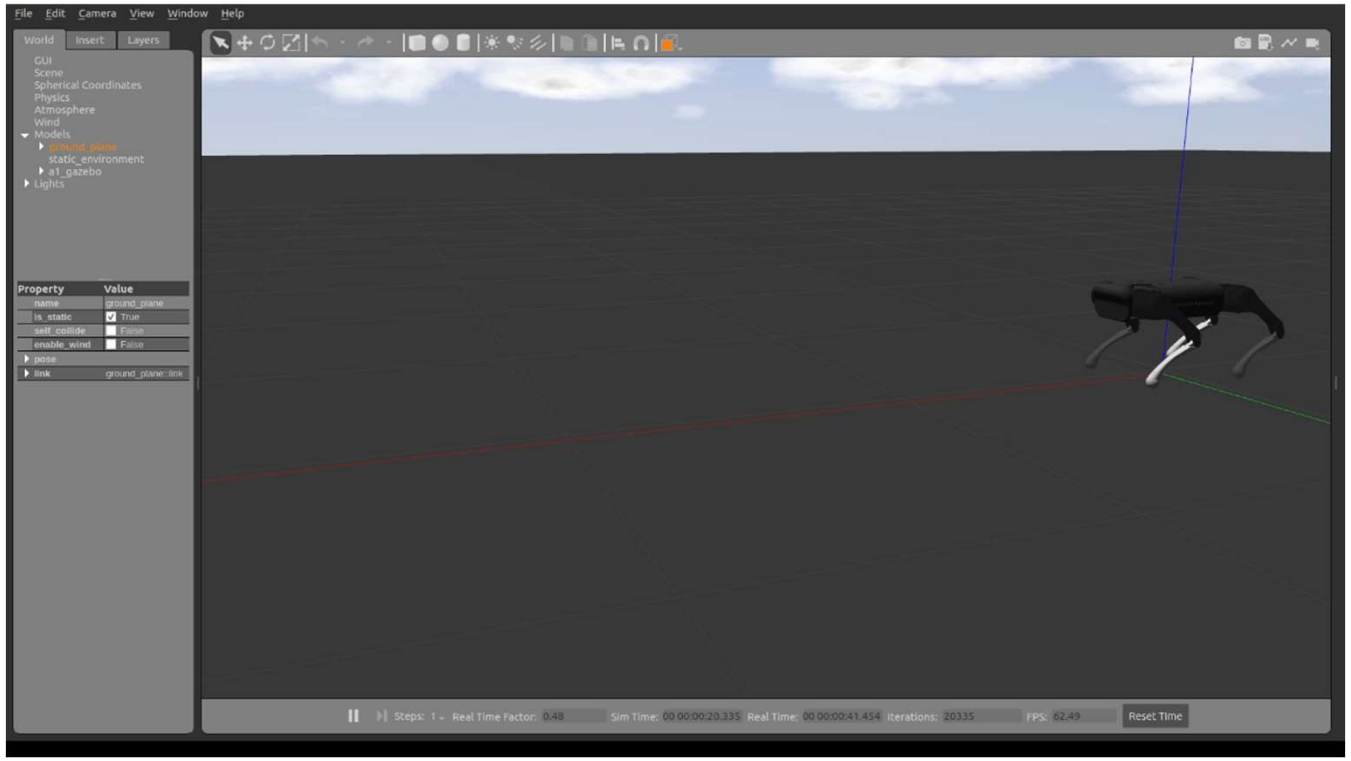
! openloop_gait_generator.yaml ●

home > dan > Scaricati > ! openloop_gait_generator.yaml

```yaml
  2   gait_params:
 40     bound:
 41       stance_duration: [ 0.3, 0.3, 0.3, 0.3]
 42       duty_factor: [ 0.5, 0.5, 0.5, 0.5]
 43       init_phase_full_cycle: [ 0., 0., 0.9, 0.9]
 44       initial_leg_state: [ 0, 0, 1, 1]
 45       contact_detection_phase_threshold: 0.1
 46     pace:
 47       stance_duration: [0.2, 0.2, 0.2, 0.2]
 48       duty_factor: [0.5, 0.5, 0.5, 0.5]
 49       init_phase_full_cycle: [1., 0., 1., 0.]
 50       initial_leg_state: [1, 0, 1, 0]
 51       contact_detection_phase_threshold: 0.1
 52     gallop:
 53       stance_duration: [0.2, 0.2, 0.2, 0.2]
 54       duty_factor: [0.5, 0.5, 0.5, 0.5]
 55       init_phase_full_cycle: [0., 0.4, 0.8, 0.2]
 56       initial_leg_state: [1, 1, 1, 1]
 57       contact_detection_phase_threshold: 0.1
 58
```

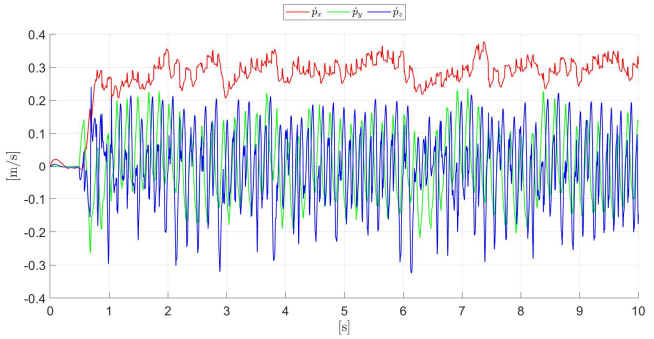
- Control tuning: $Q = [10, 10, 5, 60, 60, 100, 0., 0., 0.5, 50, 100, 1, 0]$ $R = 4 \cdot 10^{-6}\, I_{12}$

GAIT ANALYSIS – PACE

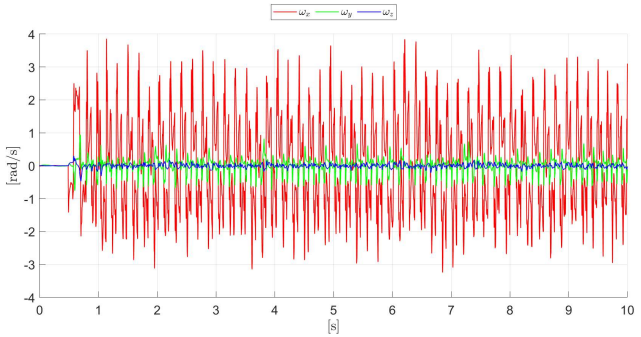


$$v_{x,d} = 0.3 \text{ m/s}$$

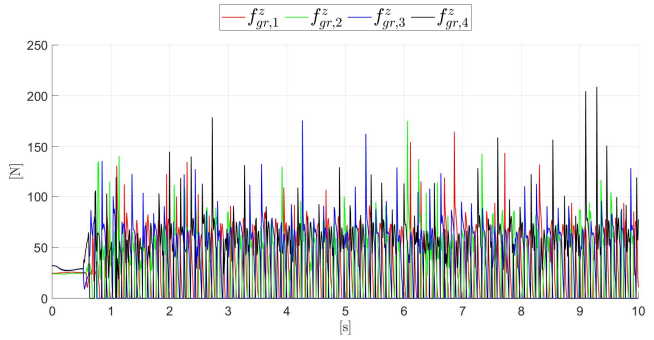
Linear Velocities



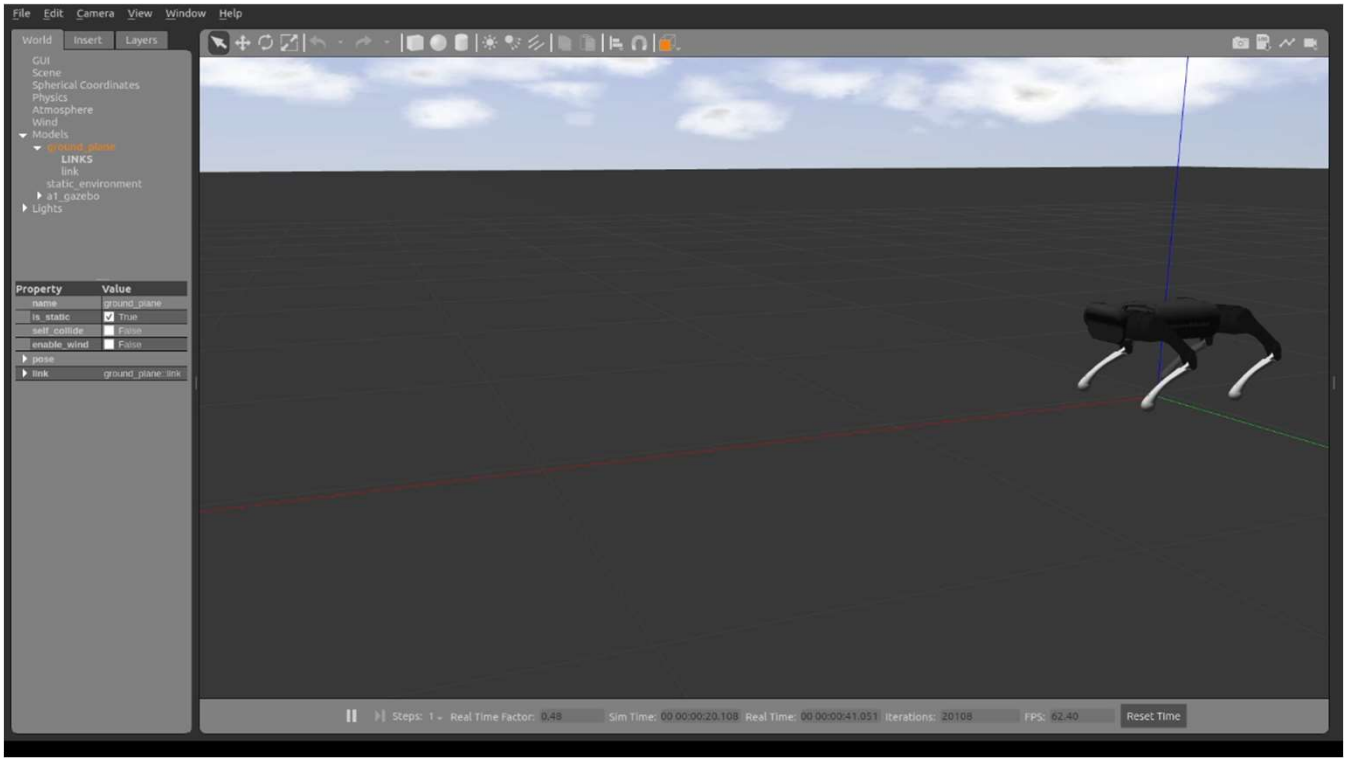
Angular Velocities



Ground Reaction Forces

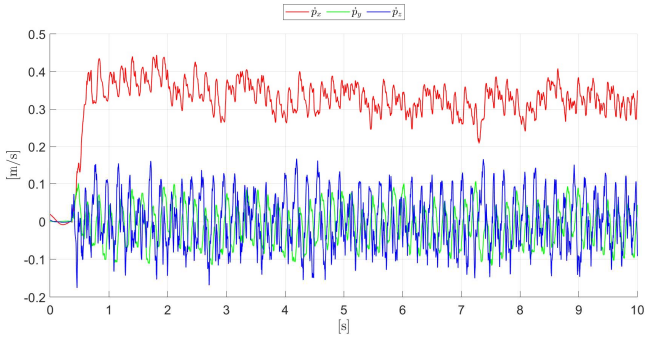


GAIT ANALYSIS – GALLOP

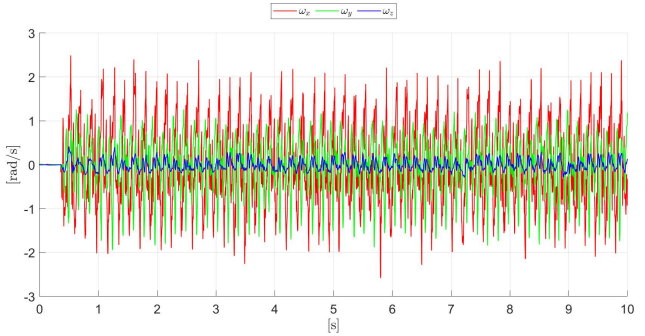


$$v_{x,d} = 0.3 \text{ m/s}$$

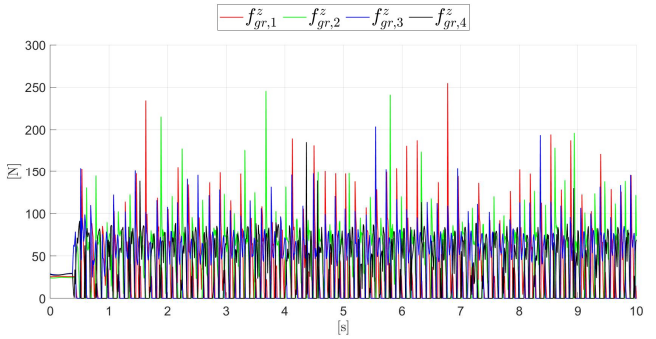
Linear Velocities



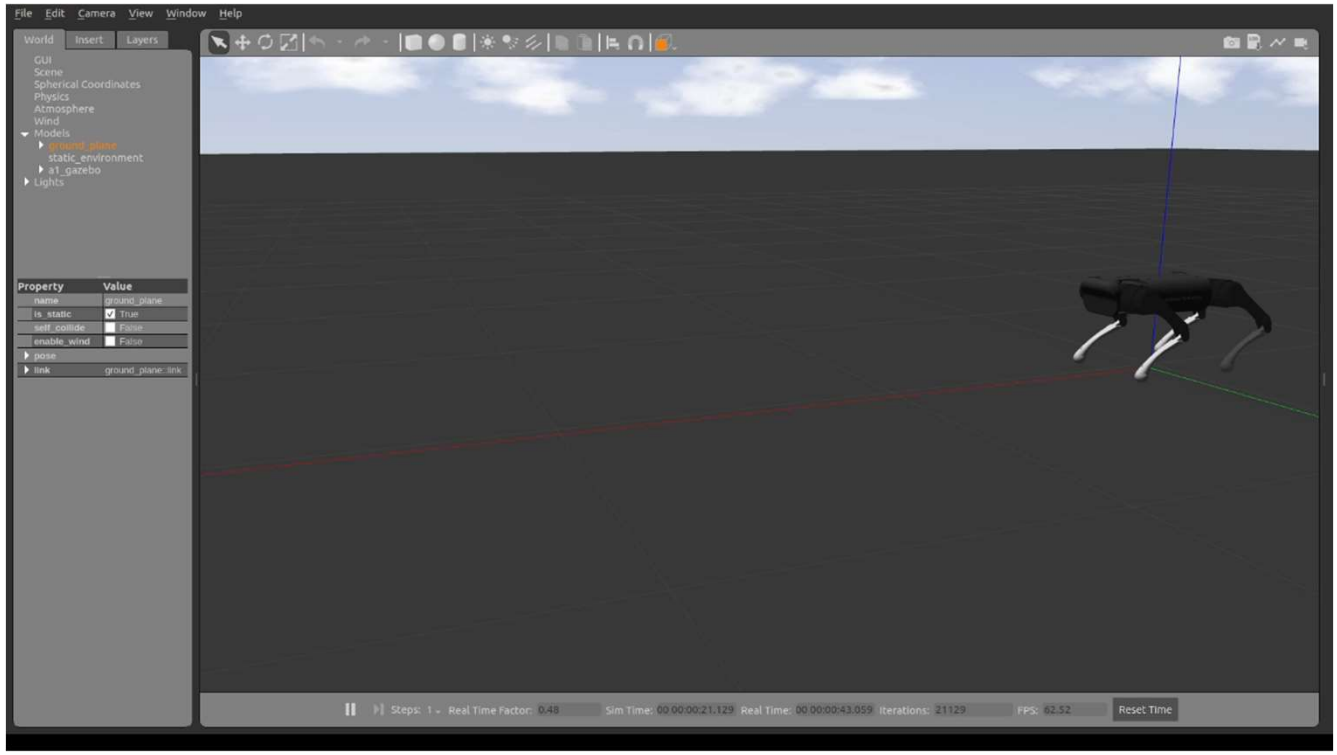
Angular Velocities



Ground Reaction Forces

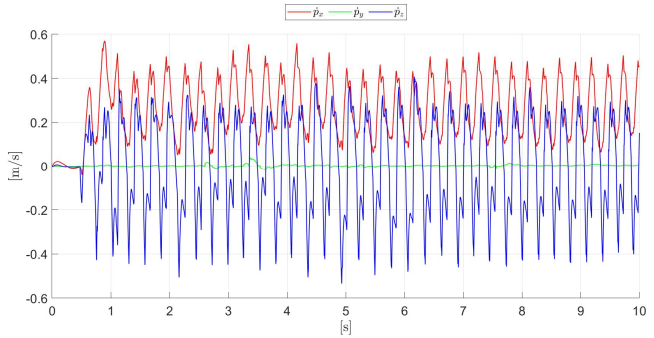


GAIT ANALYSIS – NEW BOUND

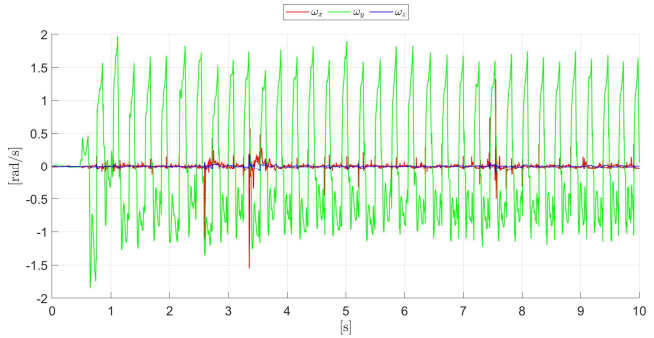


$$v_{x,d} = 0.3 \text{ m/s}$$

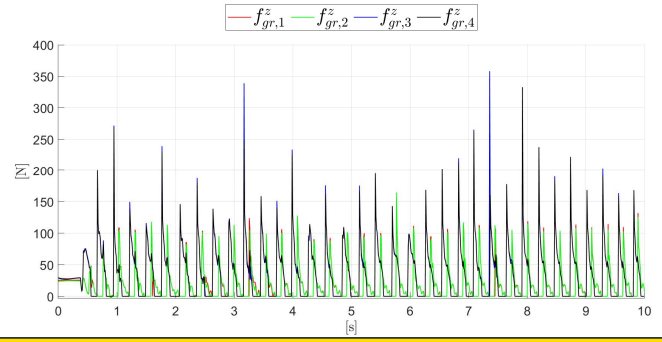
Linear Velocities



Angular Velocities

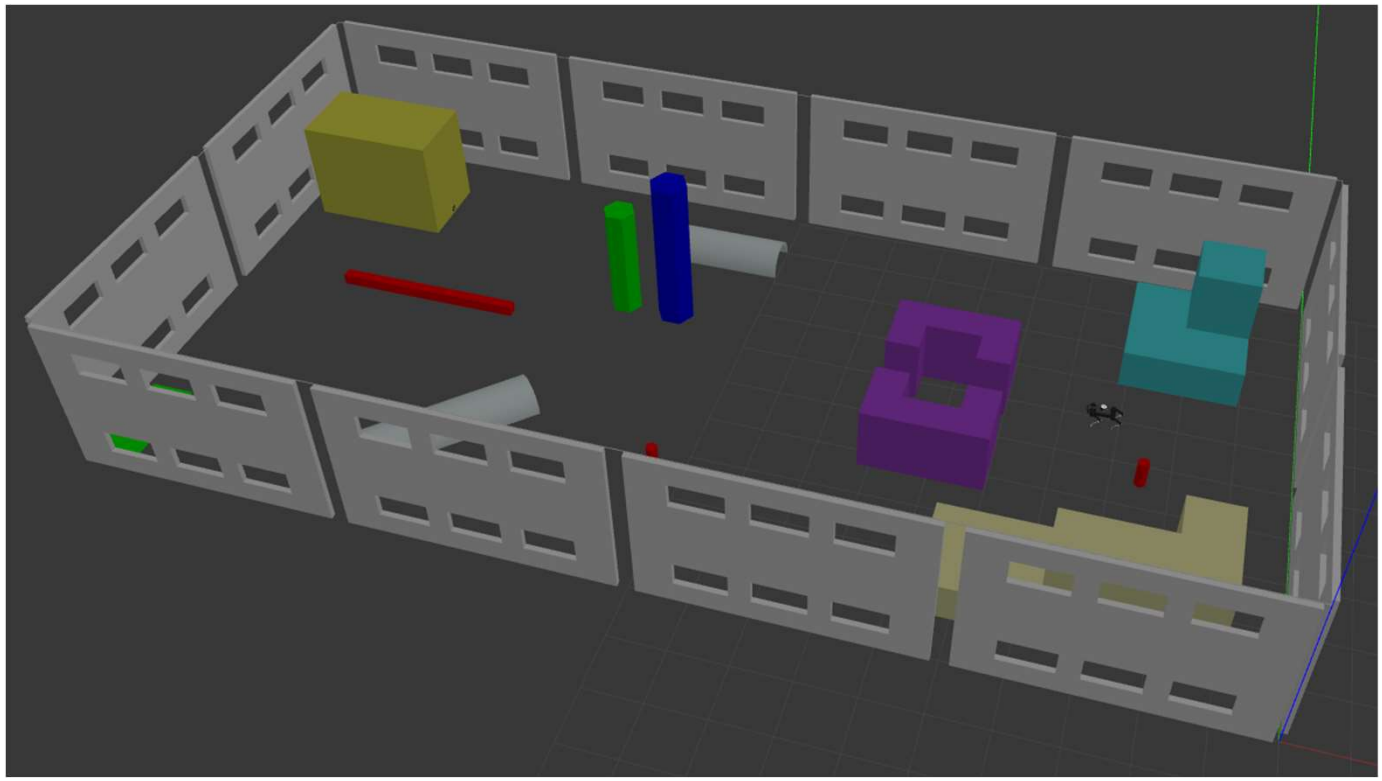


Ground Reaction Forces



RRT PATH PLANNING

- Robot moves in an environment full of obstacles

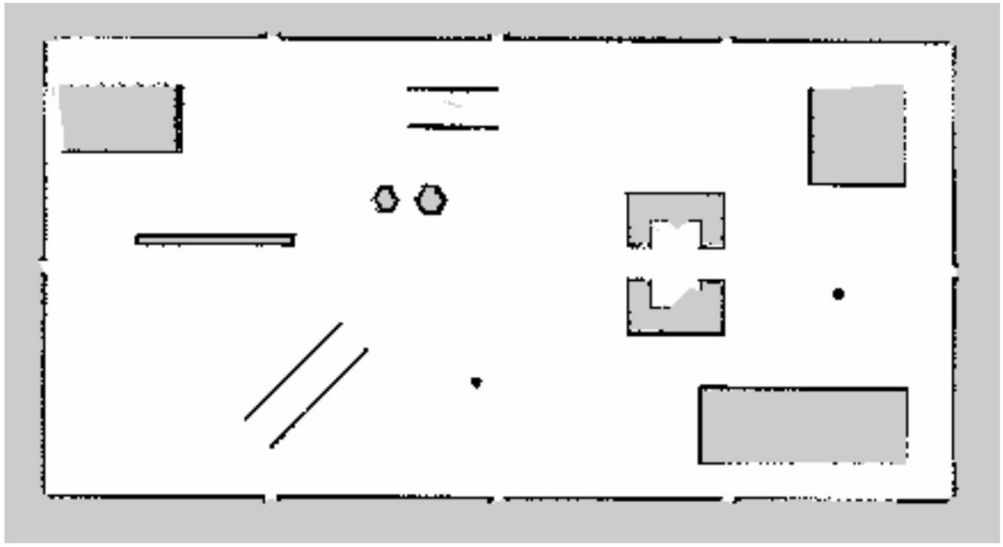


$x = -3,\ y = 6,\ z = 0.4,\ yaw = \pi$

RRT PATH PLANNING

- Set up:
  - Velodyne VLP-16 3D lidar sensor → 2D Hokuyo Laser Scanner
  - Performance improvement: < *real_time_update_rate* > = *100* (rl_race_field.world)

- Build the map using gmapping:



RRT PATH PLANNING

Key steps:

- Provide the map to move_base through the map_server node
- Start the move_base node, specifying local (DWA) and global (RRT) planners
- Start the Adaptive Monte Carlo Localization (AMCL) node to localize the robot
- Launch RVIZ with a proper configuration to visualize the RRT planner

RRT implementation:

- RRT_code.py → planner algorithm
- RRT_server.py → planner available to the move_base node
- treeviz.py → tree graphical visualization

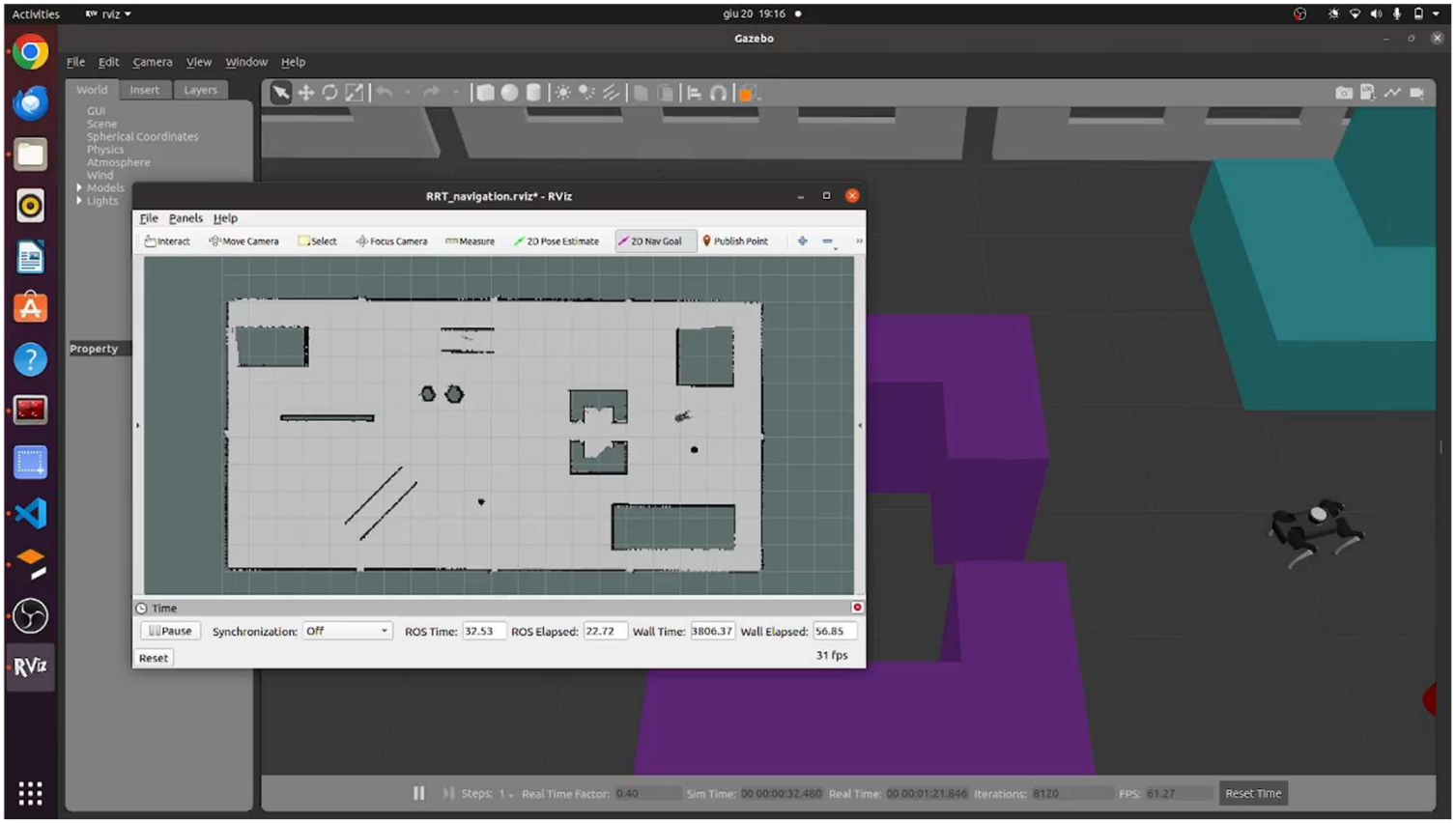
RRT PATH PLANNING

$$\delta = 10$$

$$\max\_iterations = 50$$

$$\max\_loop = 100$$

$$sample\_step = 0.5$$



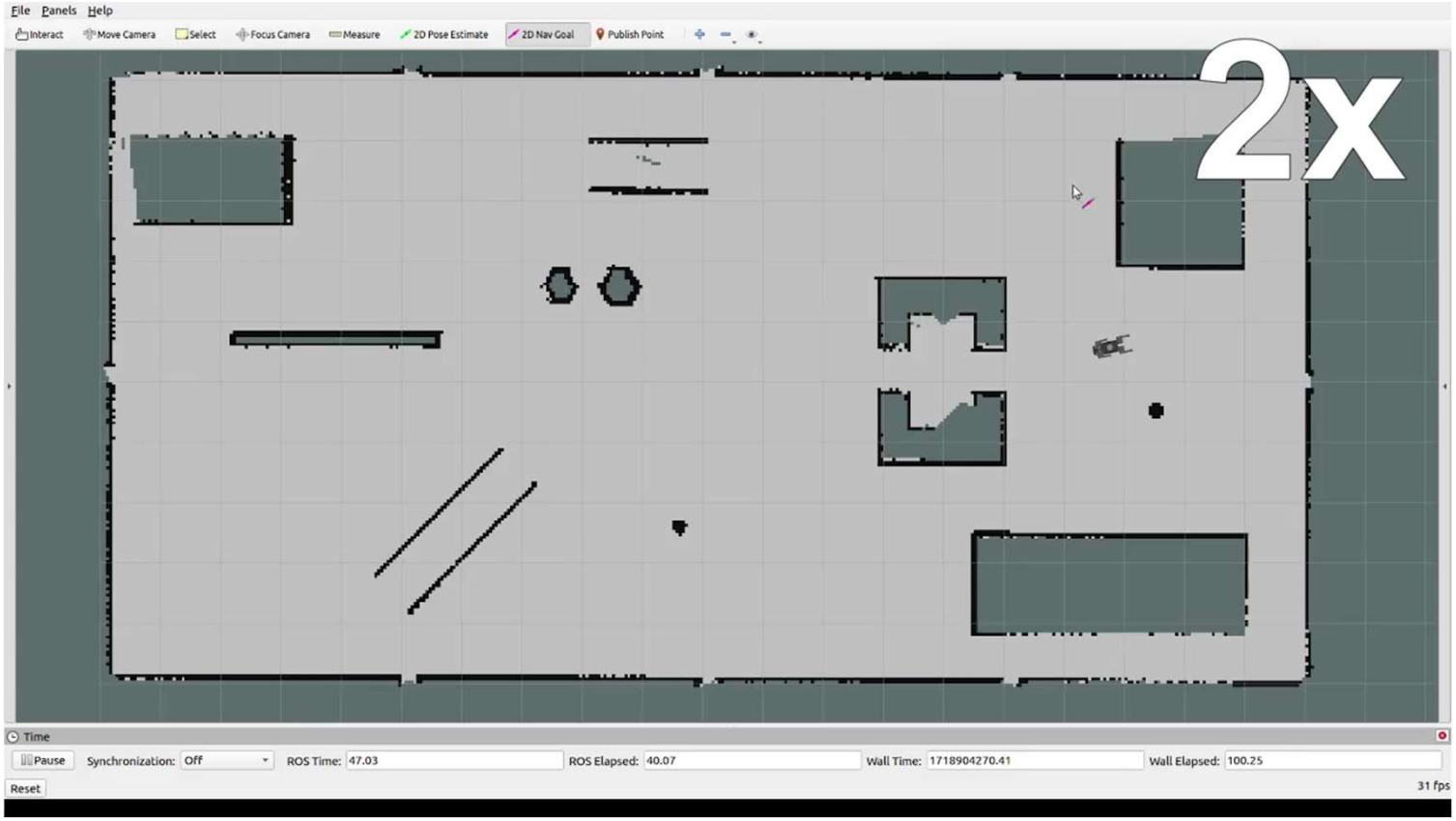
RRT PATH PLANNING

$$\delta = 10$$

$$\max\_iterations = 200$$

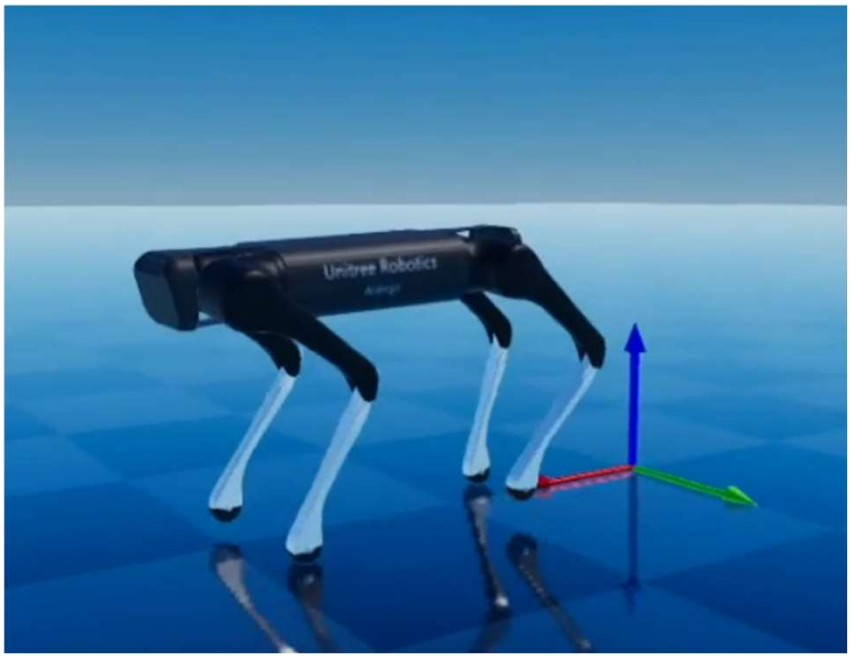
$$\max\_loop = 100$$

$$sample\_step = 0.5$$



DATA-DRIVEN CONTROL

- Deep Reinforcement Learning for controlling a quadruped robot
- Unitree Aliengo Robot
- RaiSim simulator
- Reward Machines for learning specific gaits
- Trot, Bound, Pace



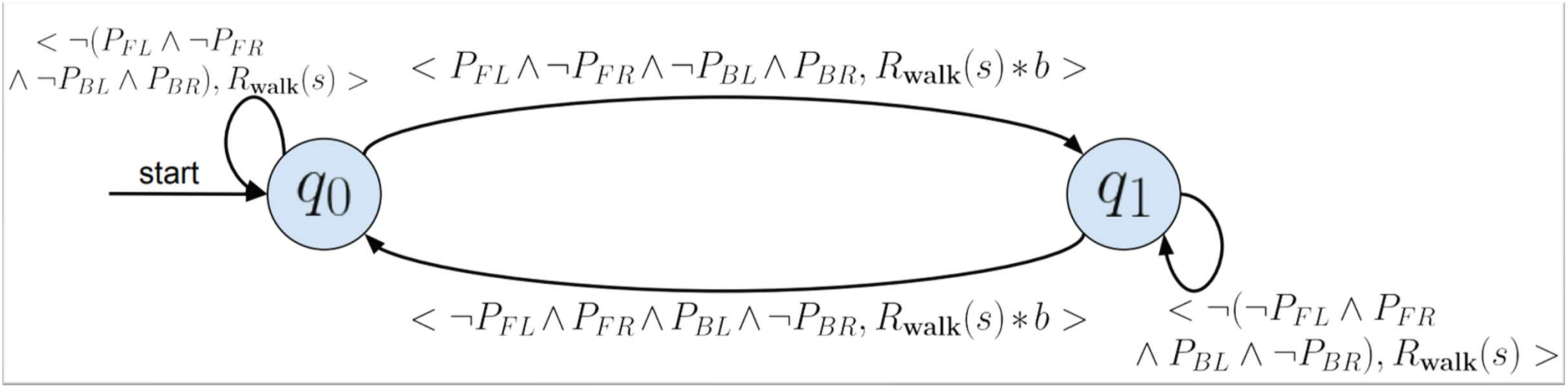
REWARD MACHINES

- Used to specify sub-goals in larger task
- Finite-state automaton
- Extend the state with higher level variables
- Still Markovian
- Can avoid the use of curriculum learning

$$RM = (U, u_0, F, \delta_u, \delta_r)$$

- U: Set of automaton states
- u_0: Initial state
- F: Set of accepting states
- δ_u: Automaton transition function
- δ_r: Automaton reward function

REWARD MACHINES



$$\delta_r(u_t, a) = \begin{cases} R_{walk}(s) + b & \text{if } \delta_u(u_t, a) \neq u_t \\ R_{walk}(s) & \text{otherwise} \end{cases}$$

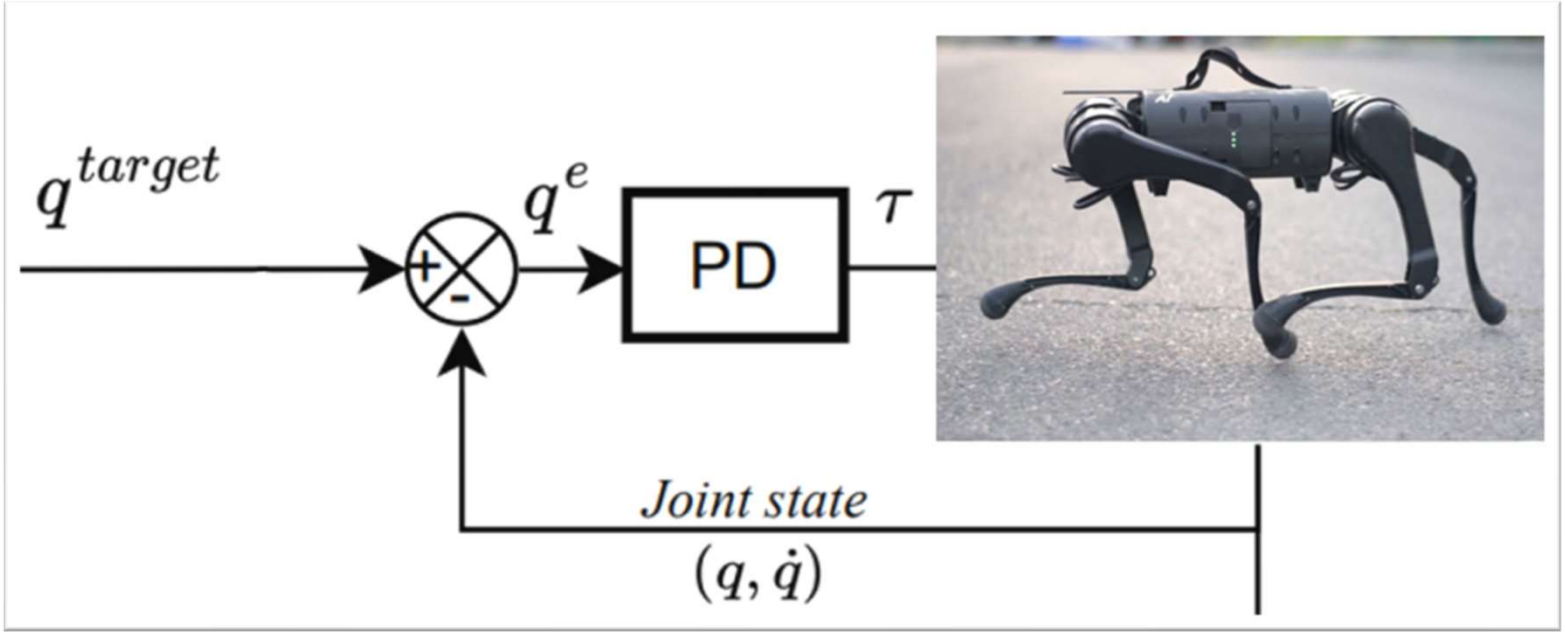
STATE SPACE

$$S = \left(q, \dot{q}, z, \phi, v, \omega, q_{t-1}^{target}, q_{t-2}^{target}, v_d, u, \delta, P\right)$$

- q: 12 joint angles
- $\dot{q}$: 12 joint velocities
- $\dot{z}$: Body height
- ϕ: Body orientation
- v: Body linear velocities
- ω: Body angular velocities
- q_{t-i}^{target}: joints target at time $t - i$
- v_d: Desired x, y velocities
- u: Current RM state
- δ: Number of steps since previous RM state change
- P: Vector of four boolean variables, that are 1 if the corresponding foot is tounching the groung

ACTION SPACE

$$\tau = K_p\left(q_t^{target} - q_t\right) - K_d\dot{q}_t \qquad K_p = 3, K_d = 0.2$$



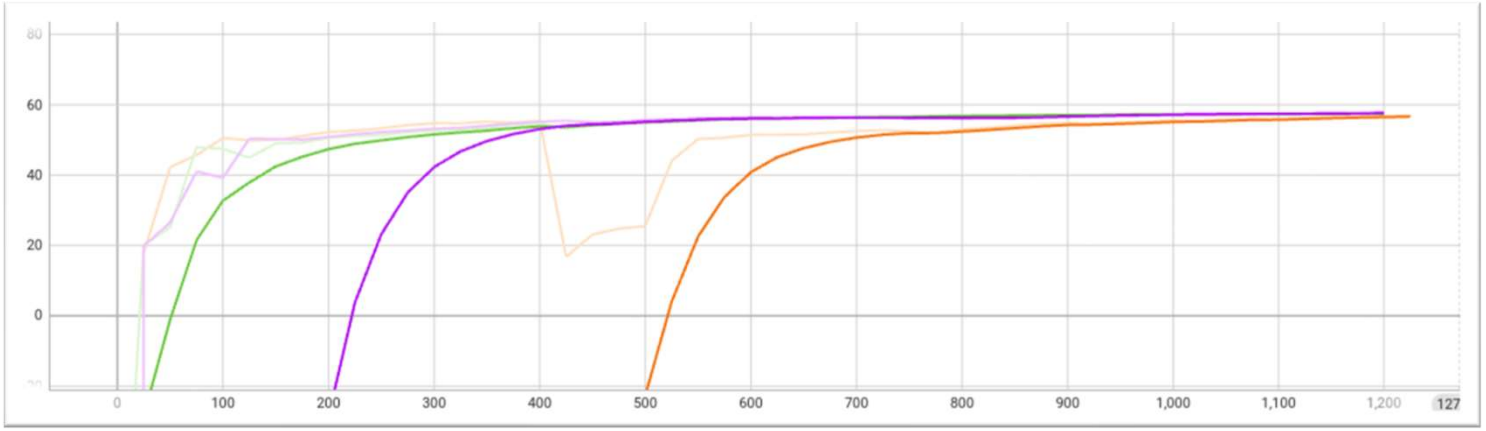
REWARD FUNCTION

Description	Equation	Weight
Linear Velocity x, y	$exp(-0.8\|\|v_{d,x,y} - v_{x,y}\|\|^2)$	10
Yaw	ψ^2	-10
Angular Velocity z	ω_z^2	-0.1
Angular Velocity x, y	$\|\|\omega_{x,y}\|\|^2$	-0.001
Joint Torques	$\|\|\tau\|\|^2$	-0.0001
Joint Positions	$\|\|q - q_{init}\|\|^2$	-0.005
Joint Velocities	$\|\|\dot{q}\|\|^2$	-0.0005
Hip Positions	$\|\|q_{hip} - q_{hip,init}\|\|^2$	-1
Foot Slip	$\sum_{i=1}^{4} P_i \|\|\dot{p}_{x,y,i}\|\|^2$	-0.1
Foot Height	$\sum_{i=1}^{4} (p_{z,i} - 0.1)^2 \|\|\dot{p}_{x,y,i}\|\|^{0.5}$	-1
Action smoothness 1	$\|\|q_t^{target} - q_{t-1}^{target}\|\|^2$	-0.5
Action smoothness 2	$\|\|q_t^{target} - 2q_{t-1}^{target} + q_{t-2}^{target}\|\|^2$	-0.5
RM transition	$b = 400$	1

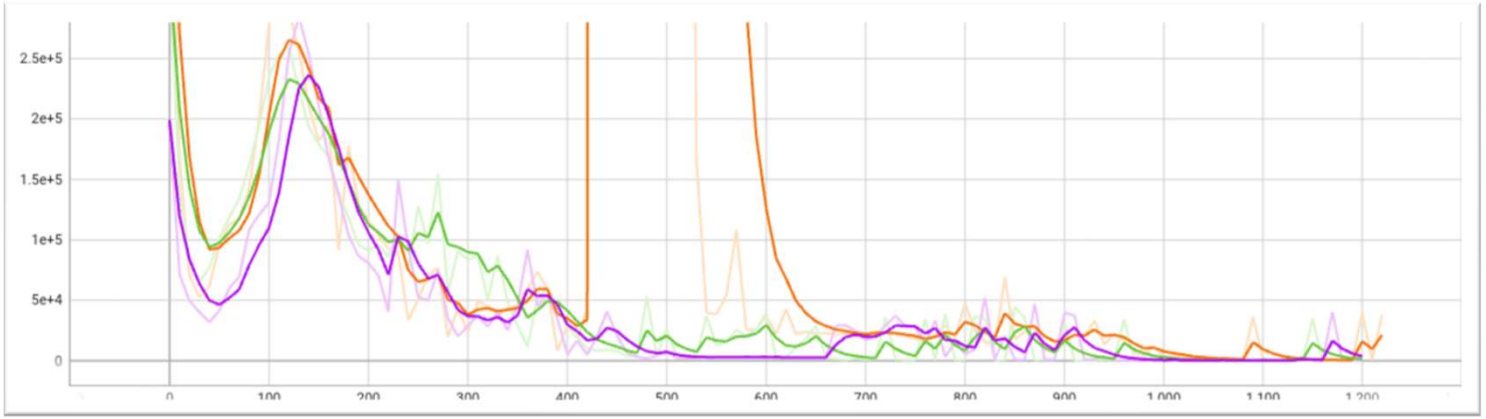
REWARD EVOLUTION

Trot Bound Pace

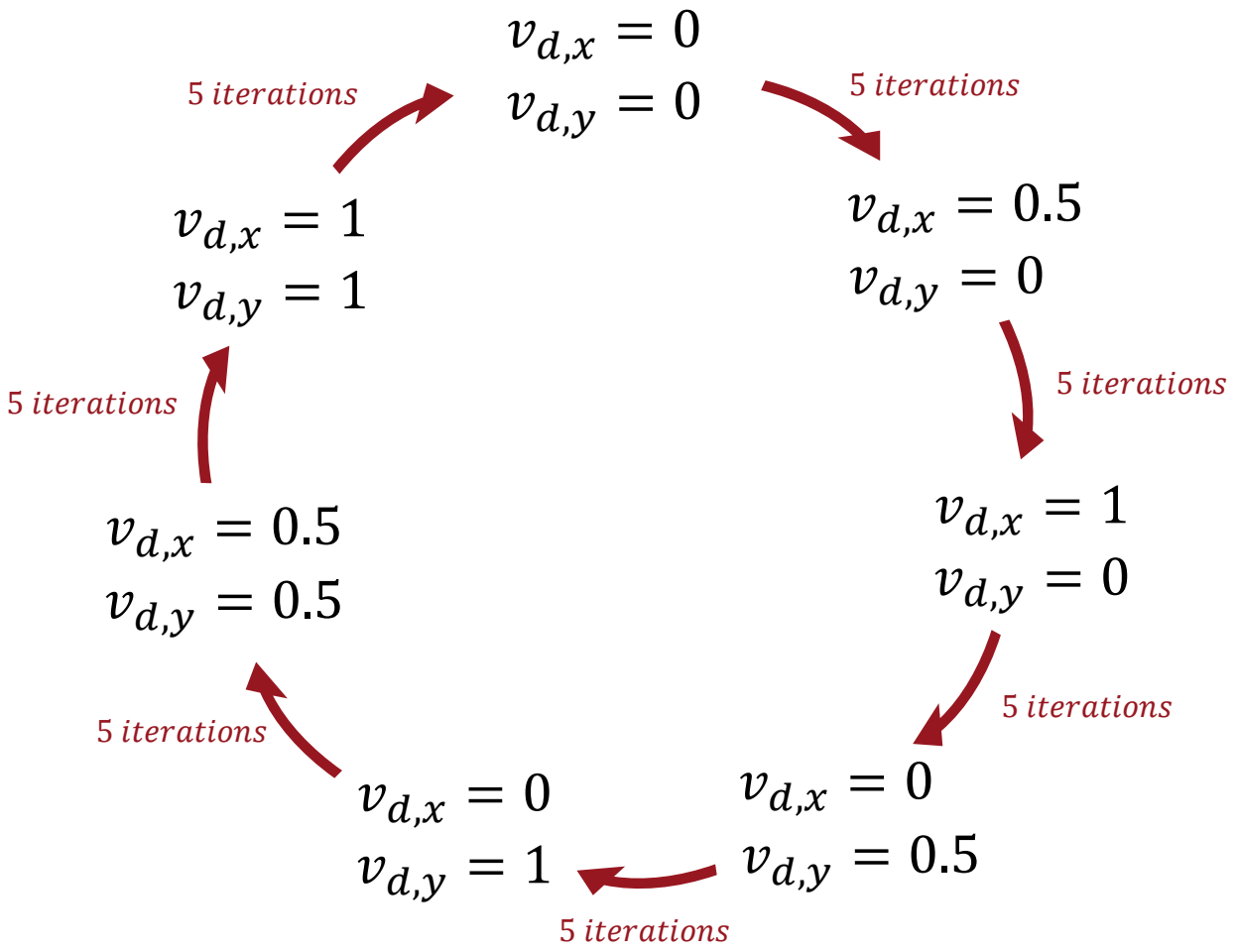
Reward mean



Loss function

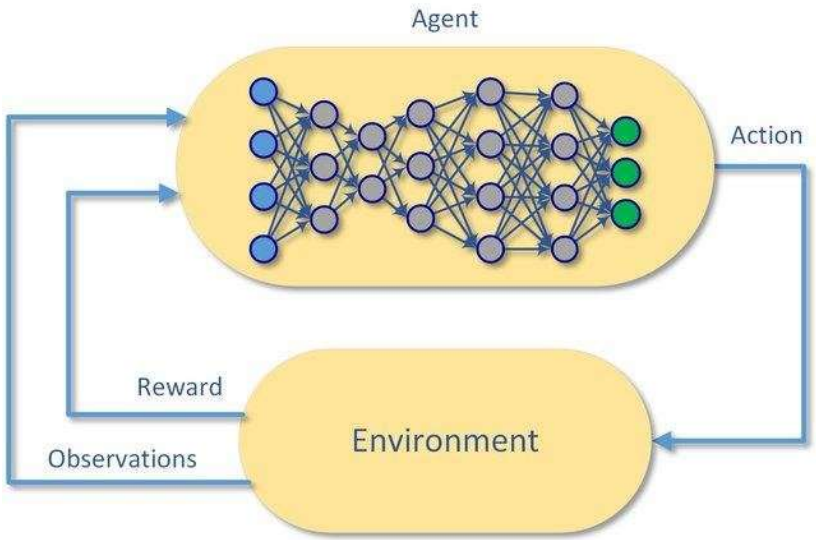


MULTIPLE DESIRED VELOCITIES TRAINING

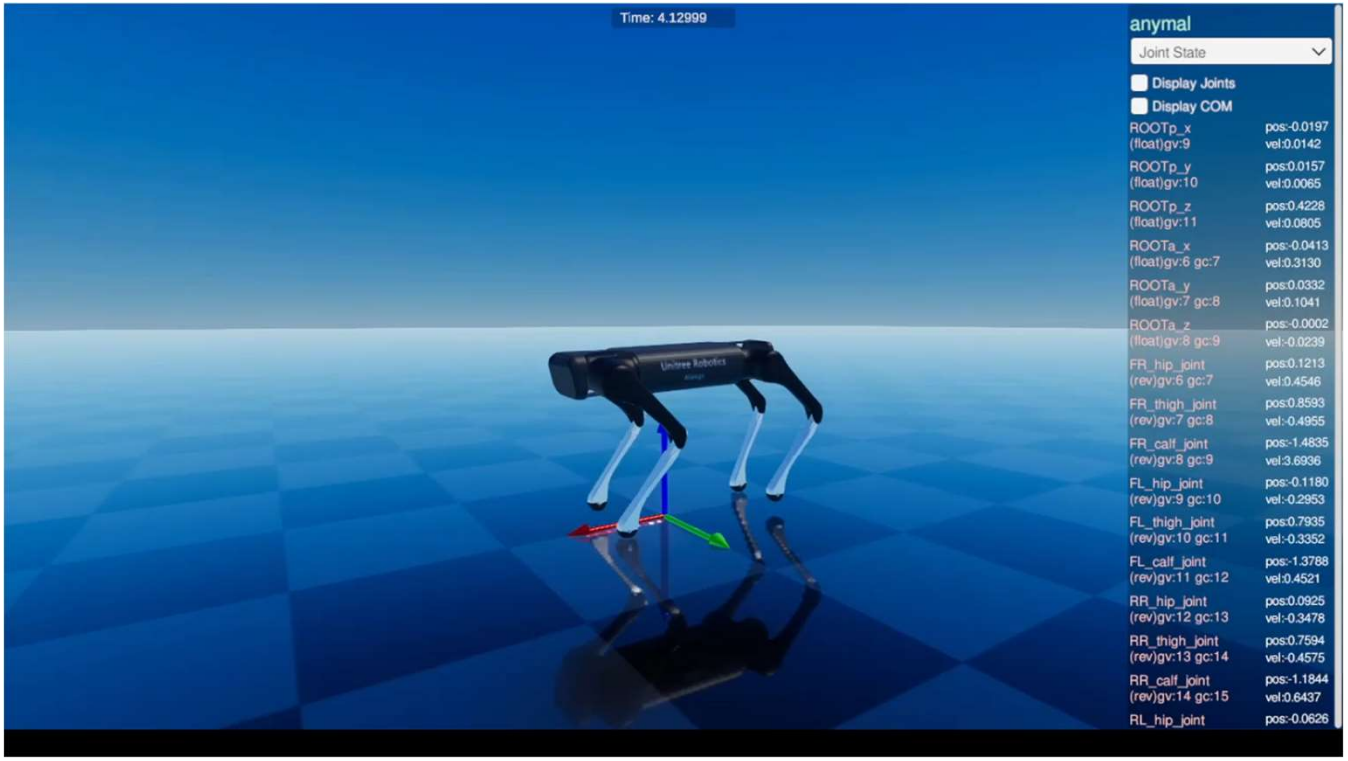


PROXIMAL POLICY OPTIMIZATION (PPO)

- Actor-critic DRL algorithm
- Multi-layer perceptron 256-128-32 neurons with Leaky ReLU
- 100 steps for each episode
- Each episode last 4 s
- 30 simulation in parallel
- Training of 1200 iteration for all three gaits
- Sampling time: 0.0025, 0.01

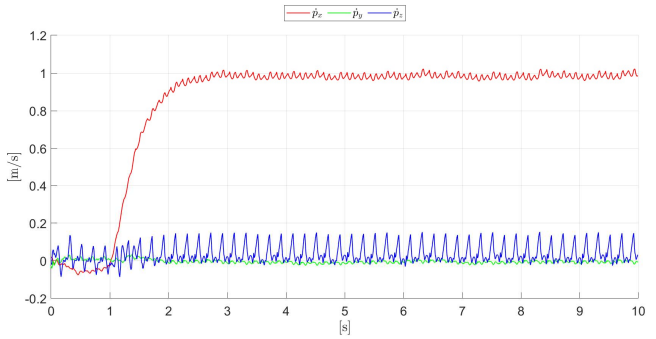


GAIT ANALYSIS – TROT

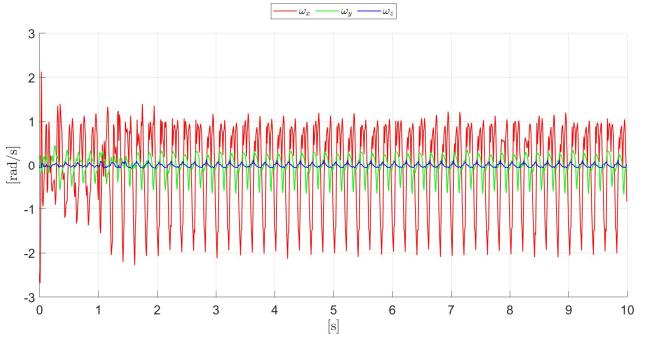


$$v_{x,d} = 1 \text{ m/s}$$

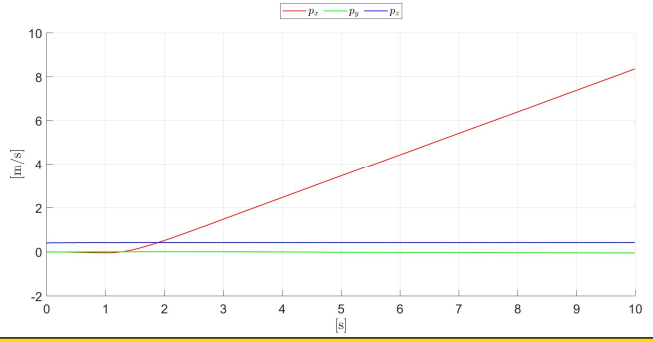
Linear Velocities



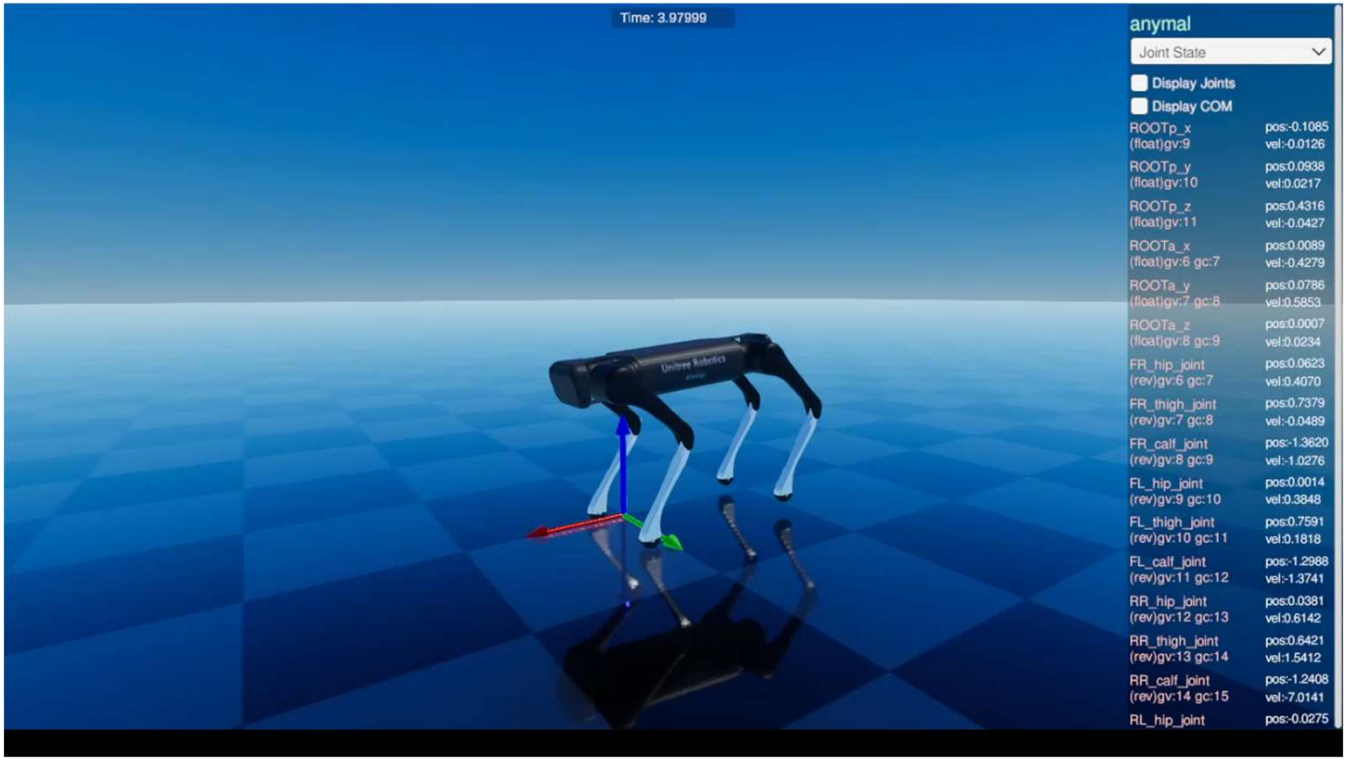
Angular Velocities



Position

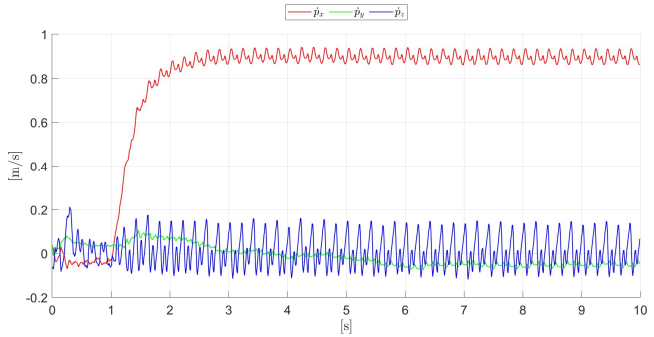


GAIT ANALYSIS – BOUND

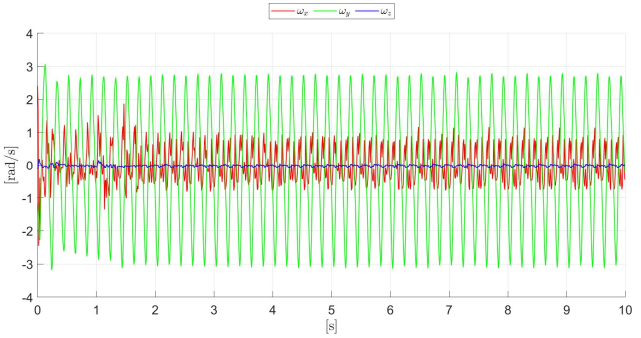


$$v_{x,d} = 1 \text{ m/s}$$

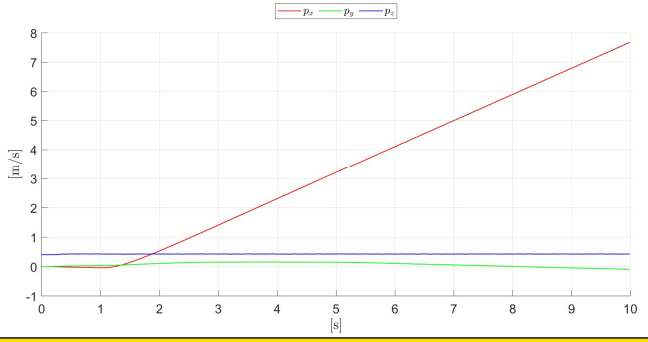
Linear Velocities



Angular Velocities



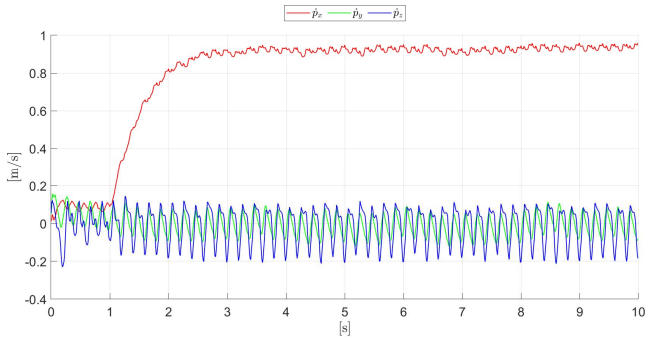
Position



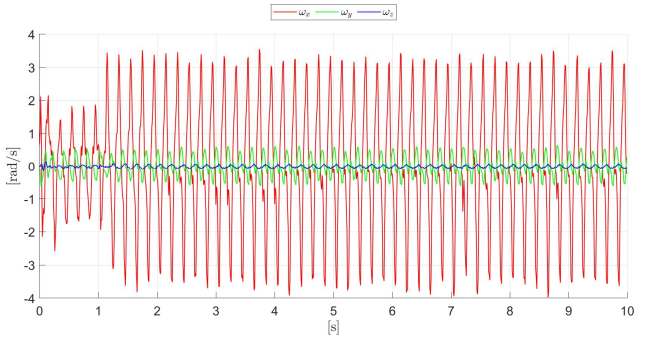
GAIT ANALYSIS – PACE

$$v_{x,d} = 1 \text{ m/s}$$

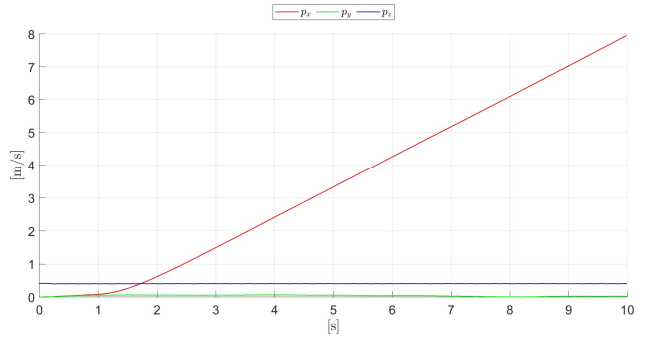
Linear Velocities



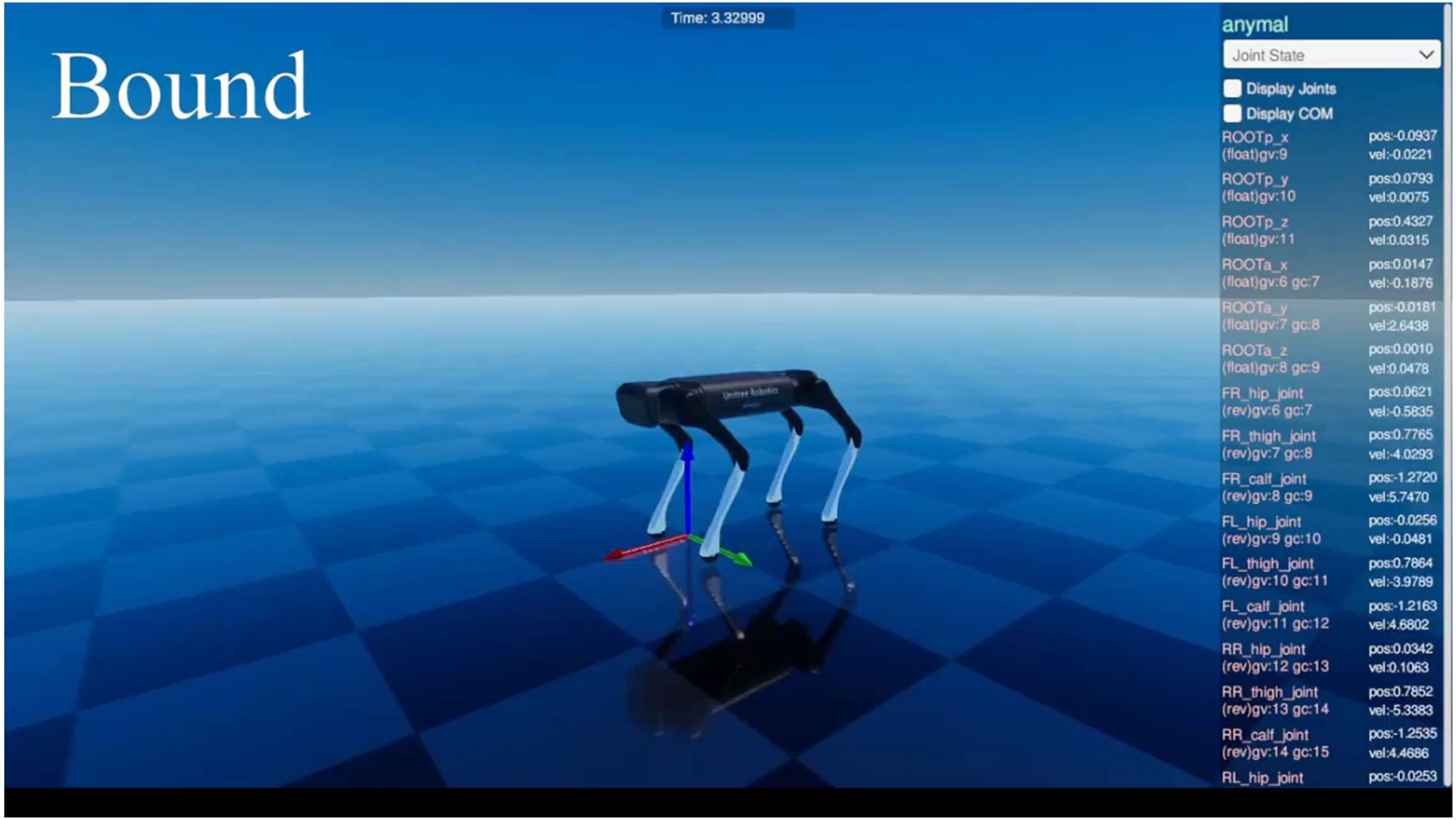
Angular Velocities



Position



EXPLOIT ALL GAITS FOR A TASK



CONCLUSIONS

Model-based Control:

- ✓ Effective with well-known system models, and clear what the control is doing
- ✓ Deterministic, easy to validate for critic applications
- X Computationally demanding, and not optimal in highly dynamic and complex environments

Data-driven Control:

- ✓ Unnecessary modeling of complex systems
- ✓ Continuous improvement with large amounts of data
- X Lots of training time and attempts, and very stochastic

THANKS FOR YOUR ATTENTION!!