Pupper project

09/06/25 update
Tund

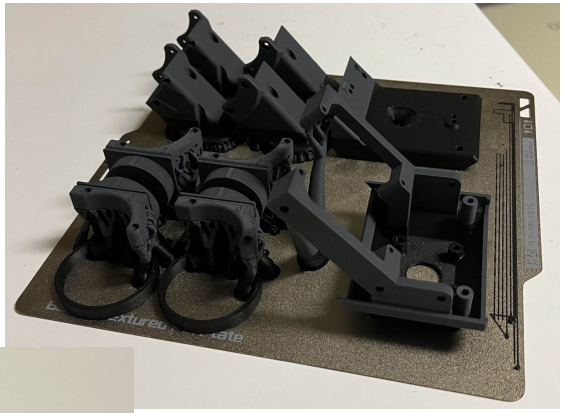
Progress

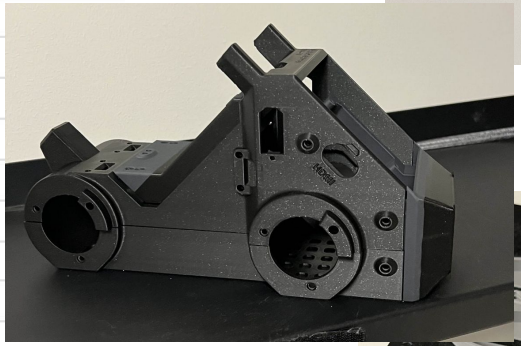
- Hardware
- Software

Hardware

3D printing ✅

Link	print status	
top half.stl	Done	Upper body
bottom h…	Done	Lower body
brain.stl	Done	Brain bracket
top tray.stl	Done	Power distribution board cover
skinny_e…	Done	Ear - RH
skinny_e…	Done	Ear - LH
front_ca…	Done	Front cable clamp - RH & LH
back_cla…	Done	Back cable clamp
strap.stl	Done	Strap
lower_le…	Done	Lower leg - RH
lower_le…	Done	Lower leg - LH
thigh_rig…	Done	Upper leg - RH
thigh_lef…	Done	Upper leg - LH
upper_le…	Done	Upper leg motor bracket RH & LH
shroud.stl	Done	Hip shroud - RH & LH
hip_brac…	Done	Hip bracket - RH
hip_brac…	Done	Hip bracket - LH





Hardware

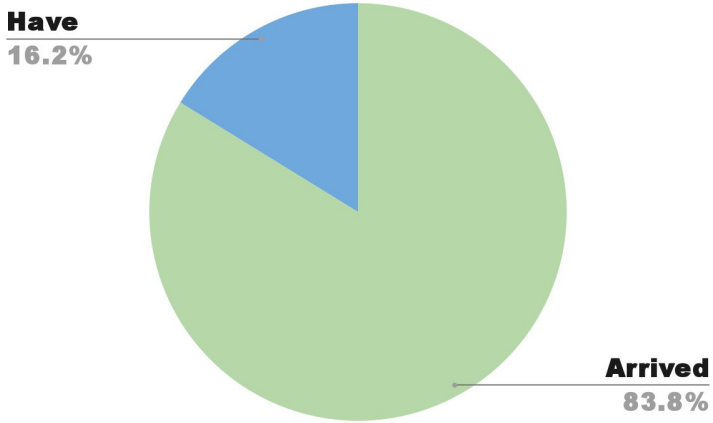
External parts

Hardware	
Have	PETG HF 1.75mm for structure
Have	M2.5x10 socket head
Arrived	M2.5 heat-set tapered threaded insert
Have	M3x8 cap head
Ordered	Squash ball single or double yellow dot
Have	[Optional] Caution hot surface label
Have	[Optional] Stand base plate

Other electronics	
Arrived	Power distribution board
Dont need	Tabs for distribution board
Ordered	Dewalt 20V powerstack battery 1.7Ah
Ordered	PS5 Wireless Controller
Ordered	[Optional] Ear servos
Have	[Optional] Wireless keyboard

Actuation	
Arrived	Steadywin GIM4305 with SHS driver

Brain electronics	
Arrived	4x4in Waveshare HDMI Raspberry Pi scr
Arrived	Rasberry Pi 5 8gb
Arrived	USB Flash drive 128 gb
Arrived	2x20 female header extension
Arrived	Pi hat control board
Ordered	[Optional] AI kit
Ordered	[Optional] M.2 adapter board for AI kit
Ordered	[Optional] Arducam IMX296
Ordered	[Optional] Fisheye lens
Ordered	[Optional] Lens holder
Dont need	[Optional] Wireless lapel microphone usb
Arrived	[Optional] Speaker



Software 

Software

Reinforcement learning

- Markov Decision Process (MDP) -
States (s): The current situation

Actions (a): The choices the agent can make

Transition (T): Probability of moving to a new state given an action

Rewards (r): Feedback signal (positive or negative)

Policy (π): Strategy that maps states to actions

Software

Software

REINFORCE - objective function

$$\nabla_\theta J(\theta) = \mathbb{E}_{\pi_\theta}[\nabla_\theta \log \pi_\theta(a_t \mid s_t) \cdot R_t]$$

$\nabla_\theta J(\theta)$: This is the gradient of the expected return with respect to policy parameters. The goal is to adjust θ to maximise return

$\log \pi_\theta(a_t \mid s_t)$: Log probability of actions given a state, under that policy

$\nabla_\theta \log \pi_\theta(a_t \mid s_t) \cdot R_t$: if the reward is positive, it will increase the probability of the action, if negative it decreases it

$\mathbb{E}_{\pi_\theta}$: Averaging over many episodes from the current policy

Cons ❌

- Increase the log-probability of actions that led to high return.
- No constraints — can make big, unstable updates.

Software

TRPO - objective function

$$\mathbb{E}_{(s,a)\sim\pi_{\theta_{\text{old}}}}\left[\frac{\pi_\theta(a \mid s)}{\pi_{\theta_{\text{old}}}(a \mid s)} \cdot \hat{A}(s,a)\right]$$

$\frac{\pi_\theta(a \mid s)}{\pi_{\theta_{\text{old}}}(a \mid s)}$: This ratio represents the fractional change in the probability of taking action when the policy changes from old parameters to new

$\hat{A}(s,a)$: Advantage is the (current_reward - baseline) see if it better or worse than average

$\frac{\pi_\theta(a \mid s)}{\pi_{\theta_{\text{old}}}(a \mid s)} \cdot \hat{A}(s,a)$: How much the action's probability has changed and whether that change improves or worsens performance

$\mathbb{E}_{(s,a)\sim\pi_{\theta_{\text{old}}}}$: Use old data from previous policy to judge if new policy did better of worse

Restriction ⛔

$\mathbb{E}_{s\sim\pi_{\theta_{\text{old}}}}[D_{\text{KL}}(\pi_{\theta_{\text{old}}}(\cdot \mid s) \parallel \pi_\theta(\cdot \mid s))] \leq \delta$: new policy should not be too different from old policy limit the average divergence to be less than or equal to a small threshold δ

Software

PPO - objective function

$$L^{\text{PPO}}(\theta) = \mathbb{E}_t[min(\frac{\pi(a_t|s_t)}{\pi_{\theta\text{old}}(a_t|s_t)} \cdot A_t, \text{clip}(\frac{\pi(a_t|s_t)}{\pi_{\theta\text{old}}(a_t|s_t)}, 1-\varepsilon, 1+\varepsilon) \cdot A_t)]$$

$\text{clip}(\frac{\pi(a_t|s_t)}{\pi_{\theta\text{old}}(a_t|s_t)}, 1-\varepsilon, 1+\varepsilon)$: This clips the ratio so that it stays within a trust range, soft version of TRPO's trust region (prevents change from being too large or too small

$\min(\cdot,\cdot)$: Looks at both unclipped and clipped terms and takes the lowest value of the two to prevent the policy from changing too much, while still encouraging improvement.

Pros ✅

- Stable learning due to clipping
- Efficient much faster than TRPO

Cons ❌

- Clipping may block good updates
- Still can be sensitivity to hyperparameters needs tuning

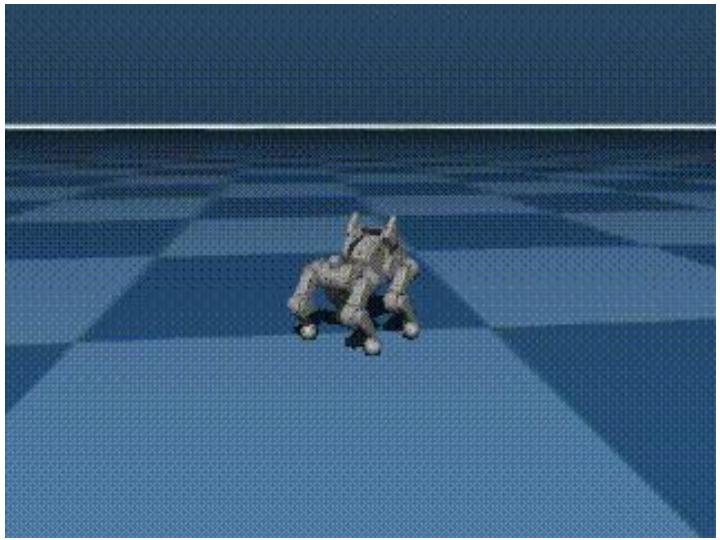
Software

Velocity tracking - naive

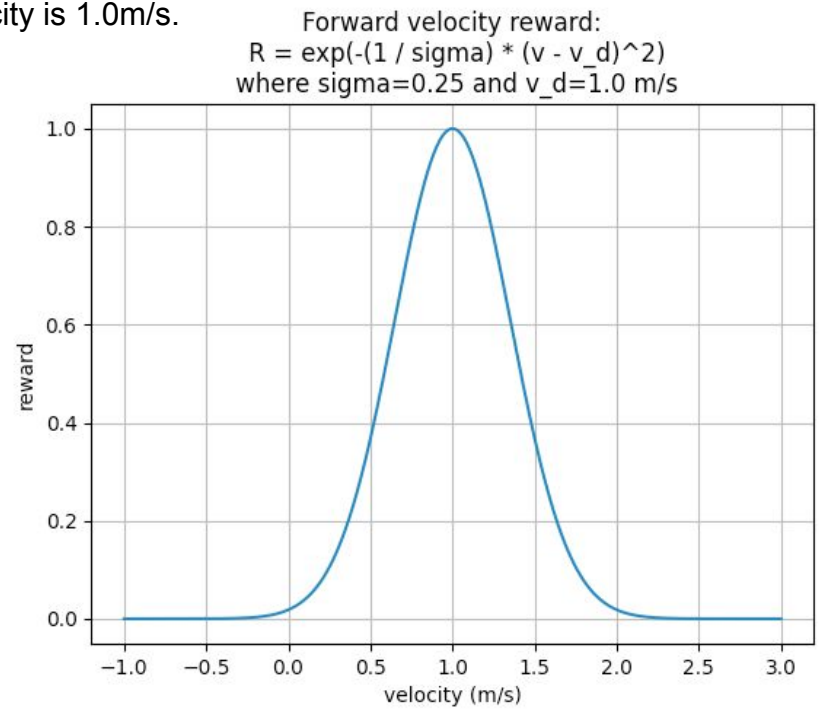
tracking_lin_vel = 2.0
tracking_ang_vel = 1.0

Why use the exponential tracking function, why not just quadratic?

- Sharp peak at target velocity
- High sensitivity near desired velocity
- Encourages precise matching of v=vdv = v_d v=vd



Exponential tracking function for Pupper to track a desired velocity. Reward value depends on the velocity of the robot when the desired x-axis velocity is 1.0m/s.



Problem ⛔

Just because Pupper has a velocity tracking reward doesn't mean it will perfectly follow the desired speed. To learn natural gaits, auxiliary rewards are needed.

Software

Effort Conservation

For Pupper to conserve effort, want to penalize excessive energy usage and unnecessary or jerky movement.

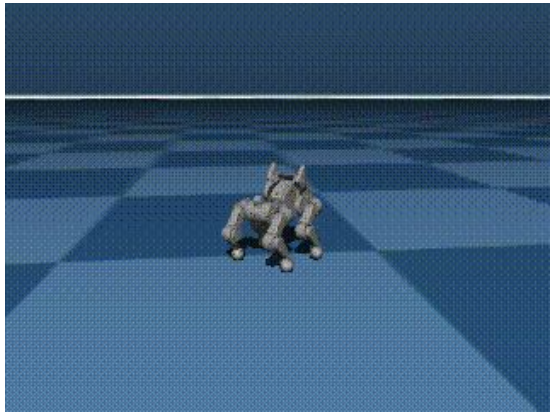
What parameters affect this:

- Joint torques
- Joint_acceleration
- Mechanical_work |v * tau|
- Action_rate |action - last_action|^2

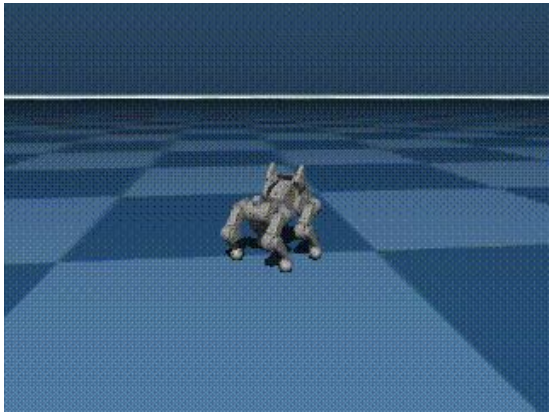
Problem ⛔

How much should we implement these (conflicting priorities)

Penalise too little



Penalise too much



Software

Effort Conservation

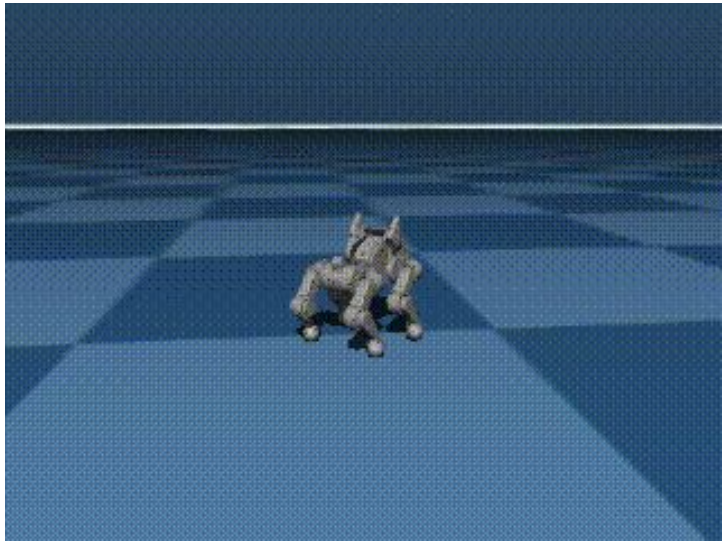
For Pupper to conserve effort, want to penalize excessive energy usage and unnecessary or jerky movement.

What parameters affect this:

- Joint torques
- Joint_acceleration
- Mechanical_work |v * tau|
- Action_rate |action - last_action|^2

Sweet spot ✅ - natural gate starting to form

About 10^-4 of the main priority which in this case is lin_vel



Next Steps ➡️

Next Steps ⬆️

Next Steps ➡️ Software

Reward tuning ⚙️

- Tune the config to make Pupper smoothly follow velocities with a natural gait
- Increase num_timesteps to at least 300 million
- aim to train a stable policy up to 0.75 m/s in simulation
- Get Pupper to be able to stand still when given 0 command

Domain randomization 🗻🌋

- Edit the environment config to represent all the situations Pupper might encounter in the real world
- Try several magnitudes of the domain randomization terms to see what works
- Iterate many times tuning the domain randomization and rewards for the best policy possible

Hardware 🔧

- When hardware components arrives look into how to operate and modify it
- Assemble components based on the building instructions
- Summarize all the safety and best practices of using this robot.

Next Steps ➡️ Software & Hardware

Export Policy for neural_controller 🧠

- Have a good and working policy and export to json file
- Map controller to PS5 controller

Pupper project

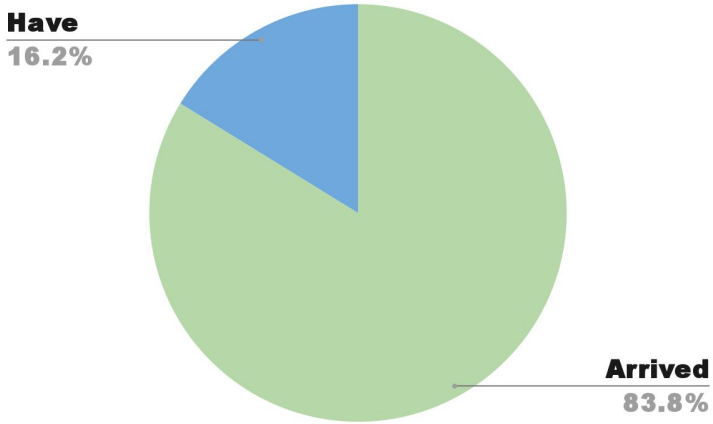
16/06/25 update
Tund

Progress

- Hardware 🔧
- Software 

Hardware

External parts ✅



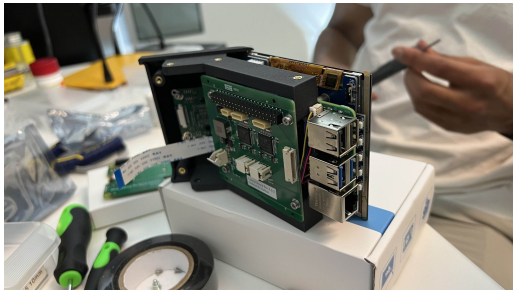
Hardware

Assembly 🛠️

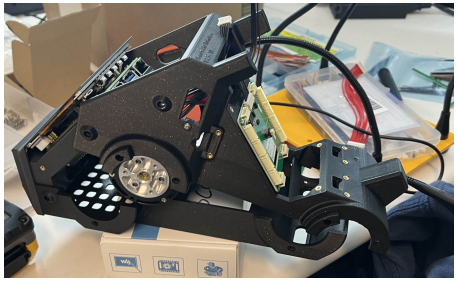
Legs 🦿

Heat threaded body 🪛

Brain 🧠



Combine 🔨



Power board ⚠️

Hardware

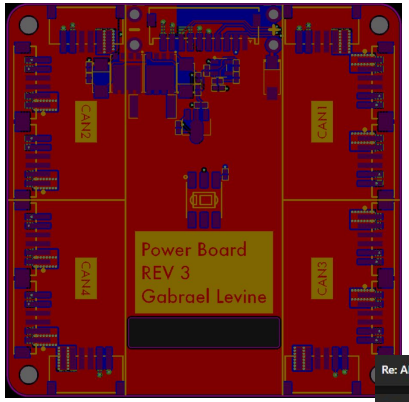
Problem 💥

Power board ⚠️

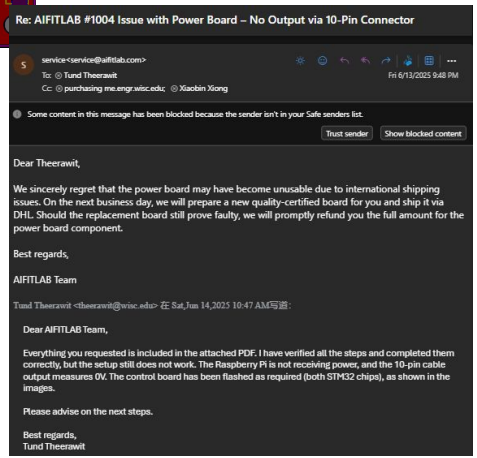
0 Power reading from 10 pin connector

Solution ✅

Order new power board



Contact manufacturer



Re: AIFITLAB #1004 Issue with Power Board – No Output via 10-Pin Connector

service<service@aifitlab.com>
To: Tund Theerawit
Cc: purchasing me.engr.wisc.edu; Xiaobin Xiong
Fri 6/13/2025 9:48 PM

Some content in this message has been blocked because the sender isn't in your Safe senders list.
Trust sender | Show blocked content

Dear Theerawit,

We sincerely regret that the power board may have become unusable due to international shipping issues. On the next business day, we will prepare a new quality-certified board for you and ship it via DHL. Should the replacement board still prove faulty, we will promptly refund you the full amount for the power board component.

Best regards,

AIFITLAB Team

Tund Theerawit <theerawit@wisc.edu> 在 Sat,Jun 14,2025 10:47 AM写道:

Dear AIFITLAB Team,

Everything you requested is included in the attached PDF. I have verified all the steps and completed them correctly, but the setup still does not work. The Raspberry Pi is not receiving power, and the 10-pin cable output measures 0V. The control board has been flashed as required (both STM32 chips), as shown in the images.

Please advise on the next steps.

Best regards,
Tund Theerawit

Software 

Software

Natural gait 

Stand still 

Next Steps ➡️

While waiting for parts to arrive 🕐

(4-7 days)

Next Steps ➡️ Hardware

- Try to assemble around the parts
- Try fix the part (anyone with pcb expertise please help 🙏)
- Use auxiliary power for raspberry pi and control board bypass power board

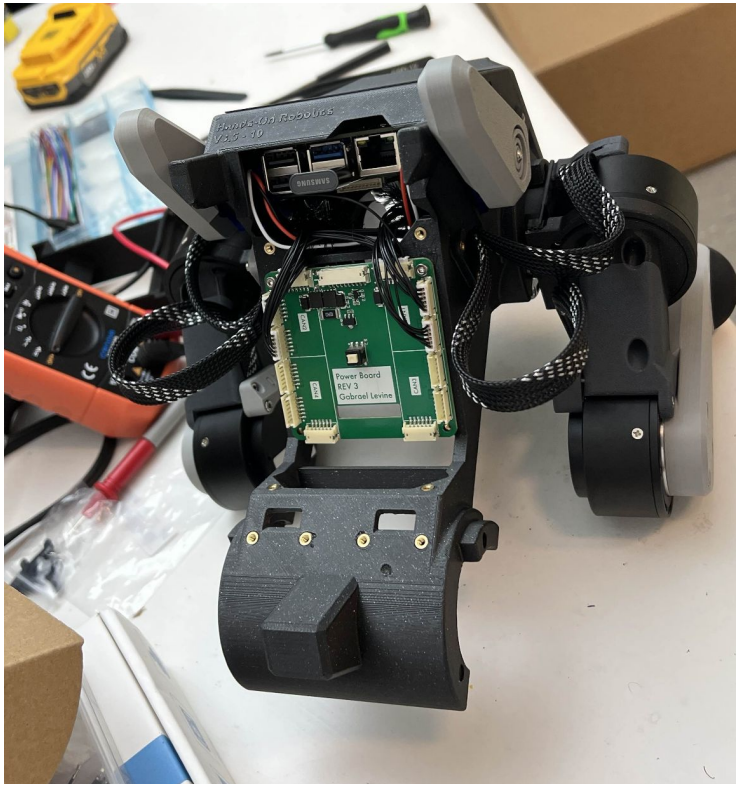
Next Steps ➡️ Software

- Try using stand still command (reward join velocity = 0
- Reward default position when stopped
- Recalculate z position

Next Steps ⬆️

While waiting for parts to arrive 🕐

(4-7 days)



Pupper (Go0.5)

Next Steps

Pupper project

30/06/25 update

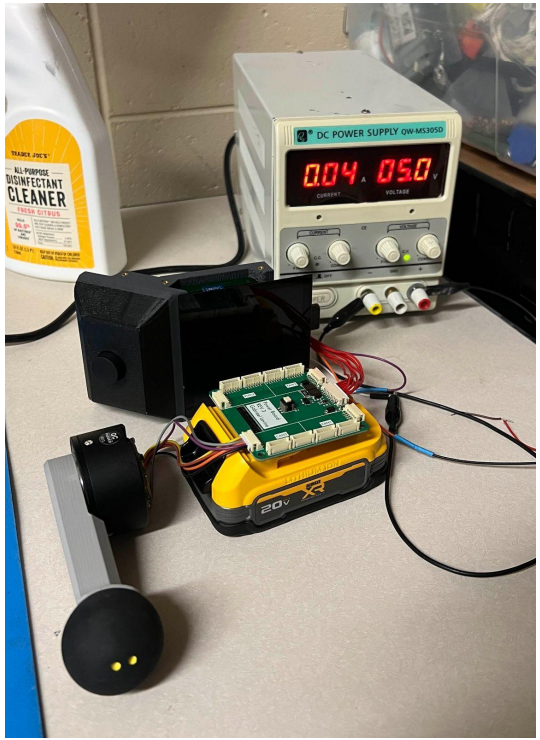
Tund

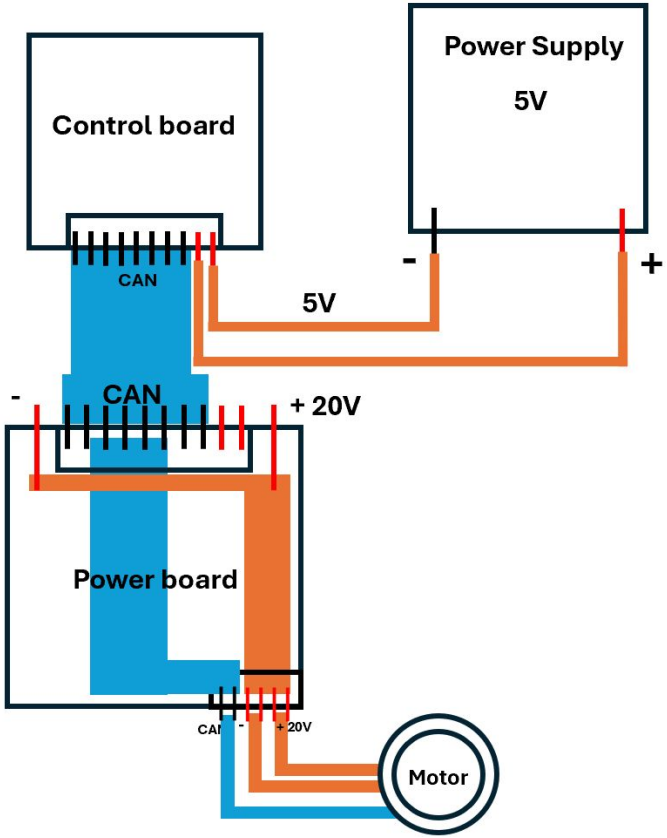
Progress

- Software 
- Hardware 

Hardware

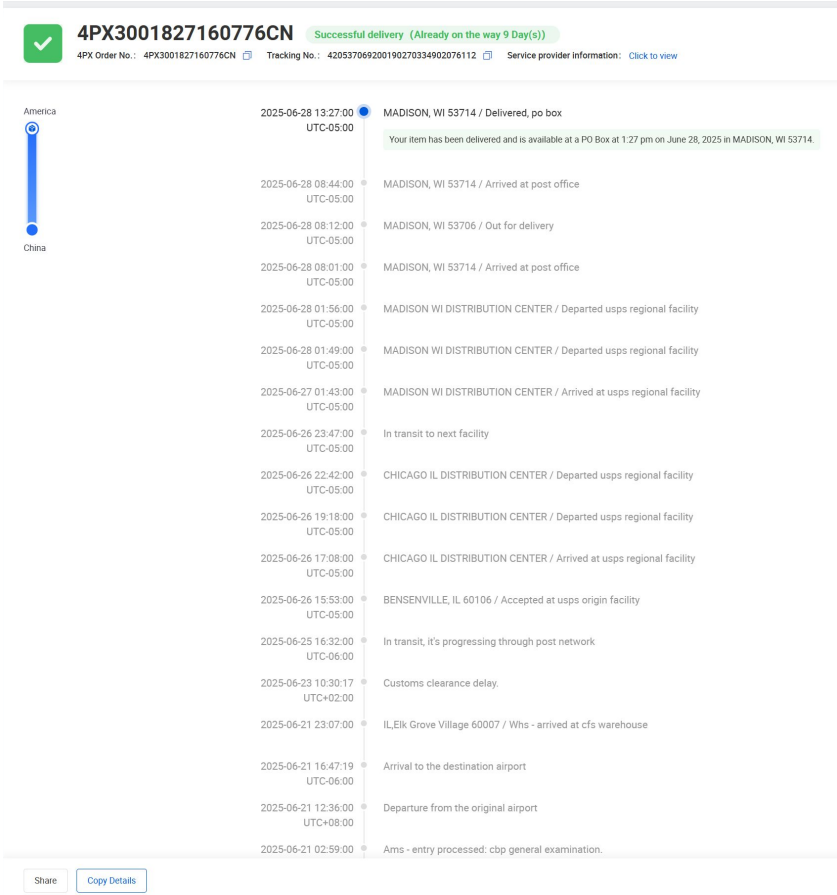
External power ⚡





Hardware

Powerboard good news ✅



4PX3001827160776CN Successful delivery (Already on the way 9 Day(s))

4PX Order No.: 4PX3001827160776CN Tracking No.: 420537069200190270334902076112 Service provider information: Click to view

America

China

2025-06-28 13:27:00 UTC-05:00	MADISON, WI 53714 / Delivered, po box Your item has been delivered and is available at a PO Box at 1:27 pm on June 28, 2025 in MADISON, WI 53714.
2025-06-28 08:44:00 UTC-05:00	MADISON, WI 53714 / Arrived at post office
2025-06-28 08:12:00 UTC-05:00	MADISON, WI 53706 / Out for delivery
2025-06-28 08:01:00 UTC-05:00	MADISON, WI 53714 / Arrived at post office
2025-06-28 01:56:00 UTC-05:00	MADISON WI DISTRIBUTION CENTER / Departed usps regional facility
2025-06-28 01:49:00 UTC-05:00	MADISON WI DISTRIBUTION CENTER / Departed usps regional facility
2025-06-27 01:43:00 UTC-05:00	MADISON WI DISTRIBUTION CENTER / Arrived at usps regional facility
2025-06-26 23:47:00 UTC-05:00	In transit to next facility
2025-06-26 22:42:00 UTC-05:00	CHICAGO IL DISTRIBUTION CENTER / Departed usps regional facility
2025-06-26 19:18:00 UTC-05:00	CHICAGO IL DISTRIBUTION CENTER / Departed usps regional facility
2025-06-26 17:08:00 UTC-05:00	CHICAGO IL DISTRIBUTION CENTER / Arrived at usps regional facility
2025-06-26 15:53:00 UTC-05:00	BENSENVILLE, IL 60106 / Accepted at usps origin facility
2025-06-25 16:32:00 UTC-06:00	In transit, it's progressing through post network
2025-06-23 10:30:17 UTC+02:00	Customs clearance delay.
2025-06-21 23:07:00	IL,Elk Grove Village 60007 / Whs - arrived at cfs warehouse
2025-06-21 16:47:19 UTC-06:00	Arrival to the destination airport
2025-06-21 12:36:00 UTC+08:00	Departure from the original airport
2025-06-21 02:59:00	Ams - entry processed: cbp general examination.

Share Copy Details

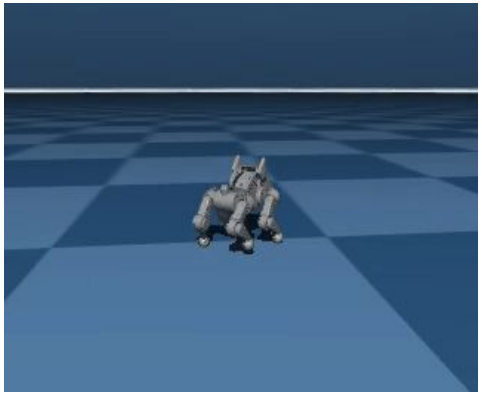
Software 

Software

PPO policy

Last week progress

Natural gait



Stand still

Software

Bayesian Optimization

Problem:

- c is defined as the reward weights
- C is $[-2,2]^n$ or some other similar bounded space (20 dimensions)
- f(c) is a scalar performance metric
  - Based on linear combinations of PPO outputs that are **expensive**, **stochastic**, and **blackbox**

$$\max_{\mathbf{c}\in\mathcal{C}} f(\mathbf{c})$$

Solution:

- Build a “surrogate” model using the Gaussian Process to estimate f(c) to determine points of interest

Process:

- Choose a small set of semi-random points and evaluate and build an initial model

$$f(\mathbf{c}) \sim \mathcal{N}(\mu(\mathbf{c}), \sigma^2(\mathbf{c}))$$

$$\begin{bmatrix} f(\mathbf{c}_1) \\ f(\mathbf{c}_2) \\ \vdots \\ f(\mathbf{c}_n) \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} \mu(\mathbf{c}_1) \\ \mu(\mathbf{c}_2) \\ \vdots \\ \mu(\mathbf{c}_n) \end{bmatrix}, \begin{bmatrix} K(\mathbf{c}_1,\mathbf{c}_1) & \dots & K(\mathbf{c}_1,\mathbf{c}_n) \\ \vdots & \ddots & \vdots \\ K(\mathbf{c}_n,\mathbf{c}_1) & \dots & K(\mathbf{c}_n,\mathbf{c}_n) \end{bmatrix} \right)$$

$$\mu(\mathbf{c}_*) = \mathbf{k}_*^\top K^{-1}\mathbf{y}$$

- The above shows the normal distribution that results from the GP

$$K(\mathbf{c}_i,\mathbf{c}_j) = \exp\left(-\frac{1}{2\ell^2}\|\mathbf{c}_i - \mathbf{c}_j\|^2\right)$$

Software

Bayesian Optimization

Process (Continued)

- Below is the Acquisition function or the Expected Improvement

$$\alpha(\mathbf{c}) = \mathbb{E}\left[\max(f(\mathbf{c}) - f_{\text{best}}, 0)\right] \qquad \alpha(\mathbf{c}) = \begin{cases} (\mu - f_{\text{best}})\Phi(Z) + \sigma\phi(Z), & \text{if } \sigma > 0 \\ 0, & \text{if } \sigma = 0 \end{cases}$$

- We pick random points in $[-2,2]^{20}$ space to compare against points already in our function.
  - Based on what is analytically determined by GP we can pick C_{next} or the next set of coeffs to try
- Once the next set is picked we retrain the policy and add the data point to the GP and then repeat the process.

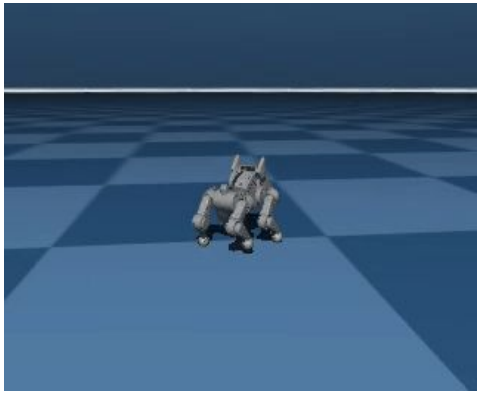
Next Steps

- Continue to work on Colab and come up with the scalar performance metric.
  - Figure how to extract useful data generated by PPO
- May put this on hold to work on redesigning pipeline for wheels and flexible limbs.

```
!pip install scikit-optimize

from skopt.space import Real
from skopt import gp_minimize


reward_config = config_dict.ConfigDict()
reward_config.rewards = config_dict.ConfigDict()
reward_config.rewards.scales = config_dict.ConfigDict()

reward_keys = list(reward_config.rewards.scales.keys())

# Create search space
reward_space = [Real(-2.0, 2.0, name=key) for key in reward_keys]

# Evaluate and reset coeffs
def eval_reward_coeffs(coeffs):
  for i, key in enumerate(reward_keys):
    reward_config.rewards.scales[key] = coeffs[i]

  config_dict_out = {key: coeffs[i] for i, key in enumerate(reward_keys)}

  with open("trial_config.json", "w") as f:
    json.dump(config_dict, f, indent=2)

res = gp_minimize(
    func=eval_reward_coeffs,
    dimensions=reward_space,
    n_calls=10,
    n_initial_points=3,
    acq_func="EI",
    random_state=42,
)
```

Software

Natural gait



Joint acceleration reward scaling

```
# L2 regularization of joint accelerations sum(|qdd|^2)
reward_config.rewards.scales.joint_acceleration = -1e-6
```

- Already quite a large number so keep reward scaling small
- Negative to discourage this behaviour

Problems ⛔

- Motion too jerky (Joint acc)
- Body too shaky (Z velocity, x&y ω, roll and pitch angles)
- Some knee collision

Take a closer look at these reward terms 👀

Joint acceleration

```
def reward_joint_acceleration(
    joint_vel: jax.Array, last_joint_vel:
jax.Array, dt: float
) -> jax.Array:
    return jp.clip(jp.sum(jp.square((joint_vel -
last_joint_vel) / (dt + EPS))), -1000.0, 1000.0)
```

- Calculates joint acceleration using finite differences
- Squares acceleration
- Sums squared accelerations across all joints

Z linear velocity

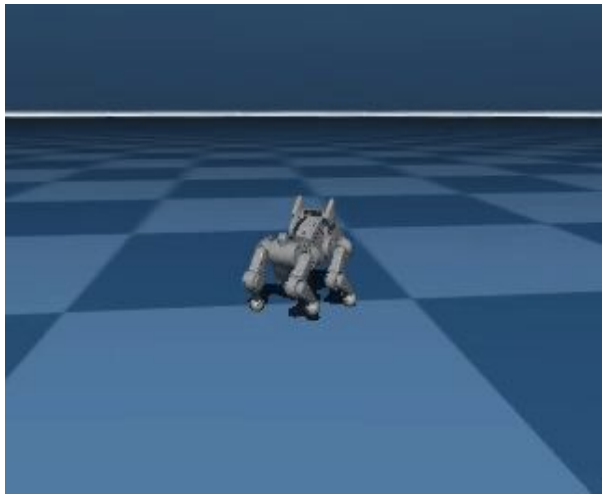
```
def reward_lin_vel_z(xd: Motion) ->
jax.Array:
    # Penalize z axis base linear
velocity
    return jp.clip(jp.square(xd.vel[0,
2]), -1000.0, 1000.0)
```

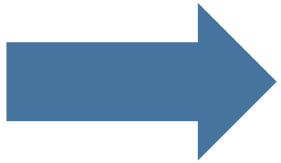
- Squares the z-velocity to ensure the penalty is always positive and to make larger velocities more heavily penalized

Software

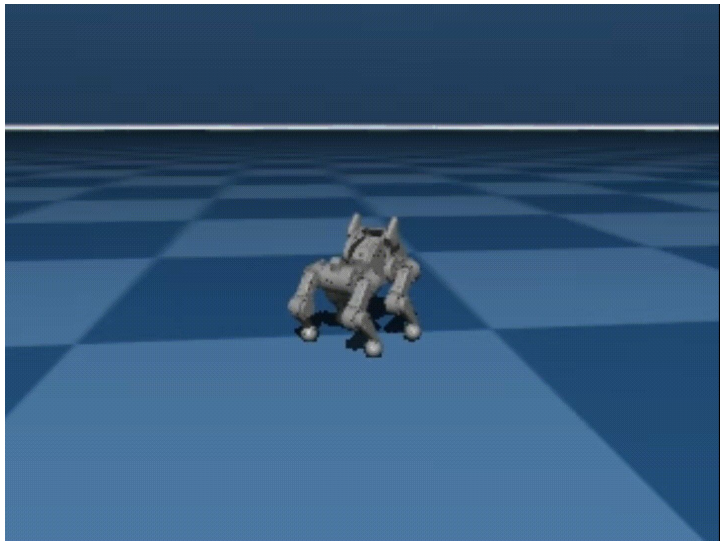
Results

Natural gait





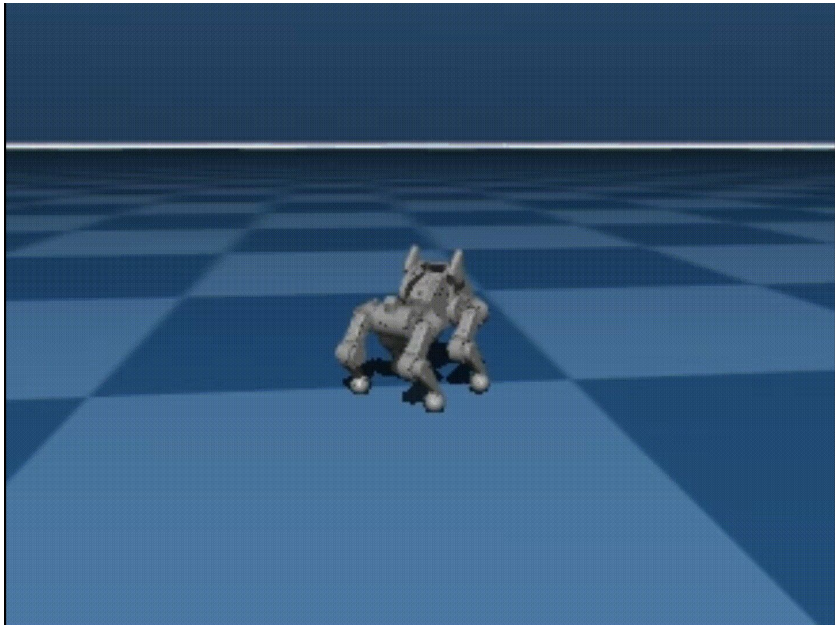
Natural gait



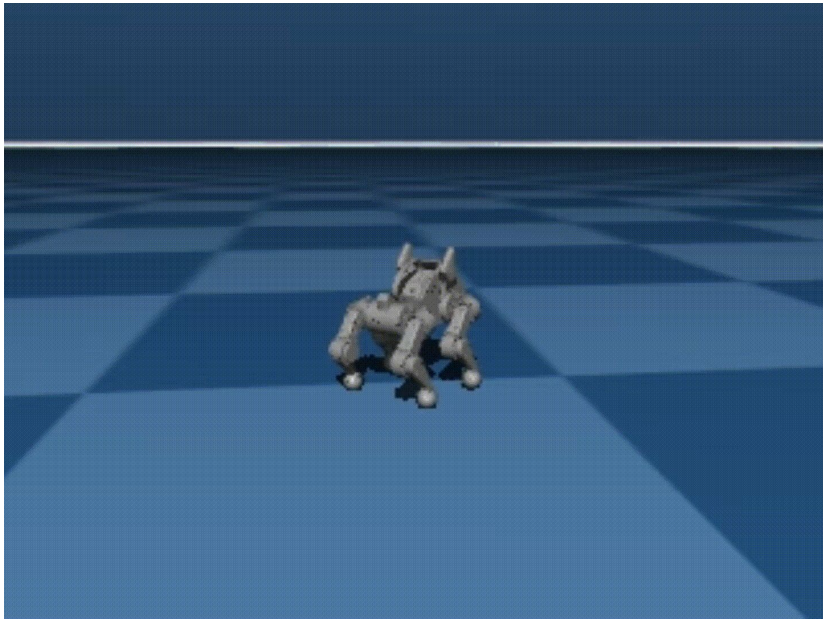
Software

Num_timesteps 200M -> 300M 🕑

200M



300M



Software

Stand still

Problems ⛔

- Doesn't stand still on 0 velocity command
- Oscillating motion to try achieve avg velocity of 0 m/s
- High joint velocity

Take a closer look at these reward terms 👀

Stand still joint velocity

```
    # Check if in stand-still mode (command magnitude below threshold)
    is_stand_still = math.normalize(commands[:3])[1] < command_threshold
    # Calculate velocity penalty (sum of squared joint velocities)
    velocity_penalty = jp.sum(jp.square(joint_velocities))

    return jp.clip(penalty, -1000.0, 1000.0)
```

- Only active when the command magnitude is below (stand-still mode).
- Uses the sum of squared joint velocities to penalize any movement.
- Ensures the reward stays within reasonable bounds.

Stand still

```
    return jp.clip(jp.sum(jp.abs(joint_angles - default_pose)) * (math.normalize(commands[:3])[1] < command_threshold),-1000.0,1000.0)
```

- Encourages the robot to maintain its default/resting pose when no movement is commanded
- Calculates the total deviation from the default pose. (Only applies the penalty when the command magnitude is below a certain threshold)

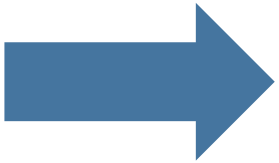
Default joint angles

```
Leg 1 (FL): [0.26, 0.0, -0.52]
Leg 2 (FR): [-0.26, 0.0, 0.52]
Leg 3 (RL): [0.26, 0.0, -0.52]
Leg 4 (RR): [-0.26, 0.0, 0.52]
```

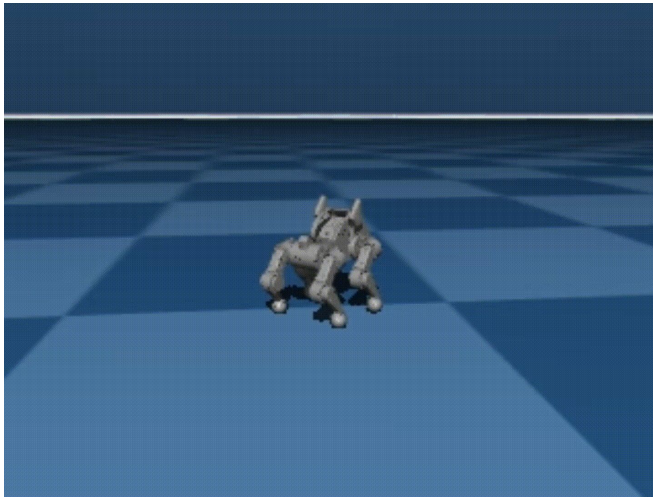
Software

Results

Stand still 



Stand still 

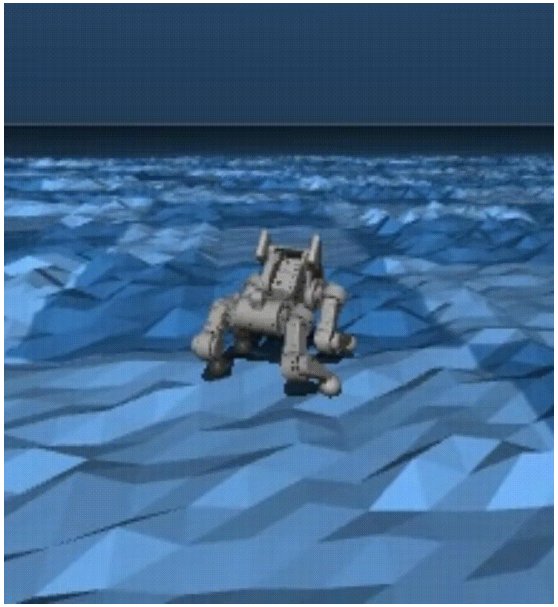


Software

Domain Randomisation 🏔️

```
# Height field
## Type of height field
training_config.height_field_random = True
# Default: false
training_config.height_field_steps = False
### Steps type params
training_config.height_field_step_size = 4
## General height field settings
training_config.height_field_grid_size =
256
training_config.height_field_group = "0"
training_config.height_field_radius_x =
10.0 # [m]
training_config.height_field_radius_y =
10.0 # [m]
training_config.height_field_elevation_z =
0.08 # [m] Def: 0.02
training_config.height_field_base_z = 0.2 #
[m]
```

Random movement



Y 0.75 (m/s)

X 0.75 (m/s)

Next Steps ➡️

Next Steps ⬆️

Next Steps ➡️ Hardware

- Fully assemble a working Pupper with the new powerboard
- Discuss smart linkage with Prof Xiao Kuang

Next Steps ➡️ Software

- Polish PPO legged policy
- Start working on PPO wheeled locomotion

Next Steps ➡️ Software + Hardware

- Export policy.json to neural controller to control Pupper using remote control

Pupper project

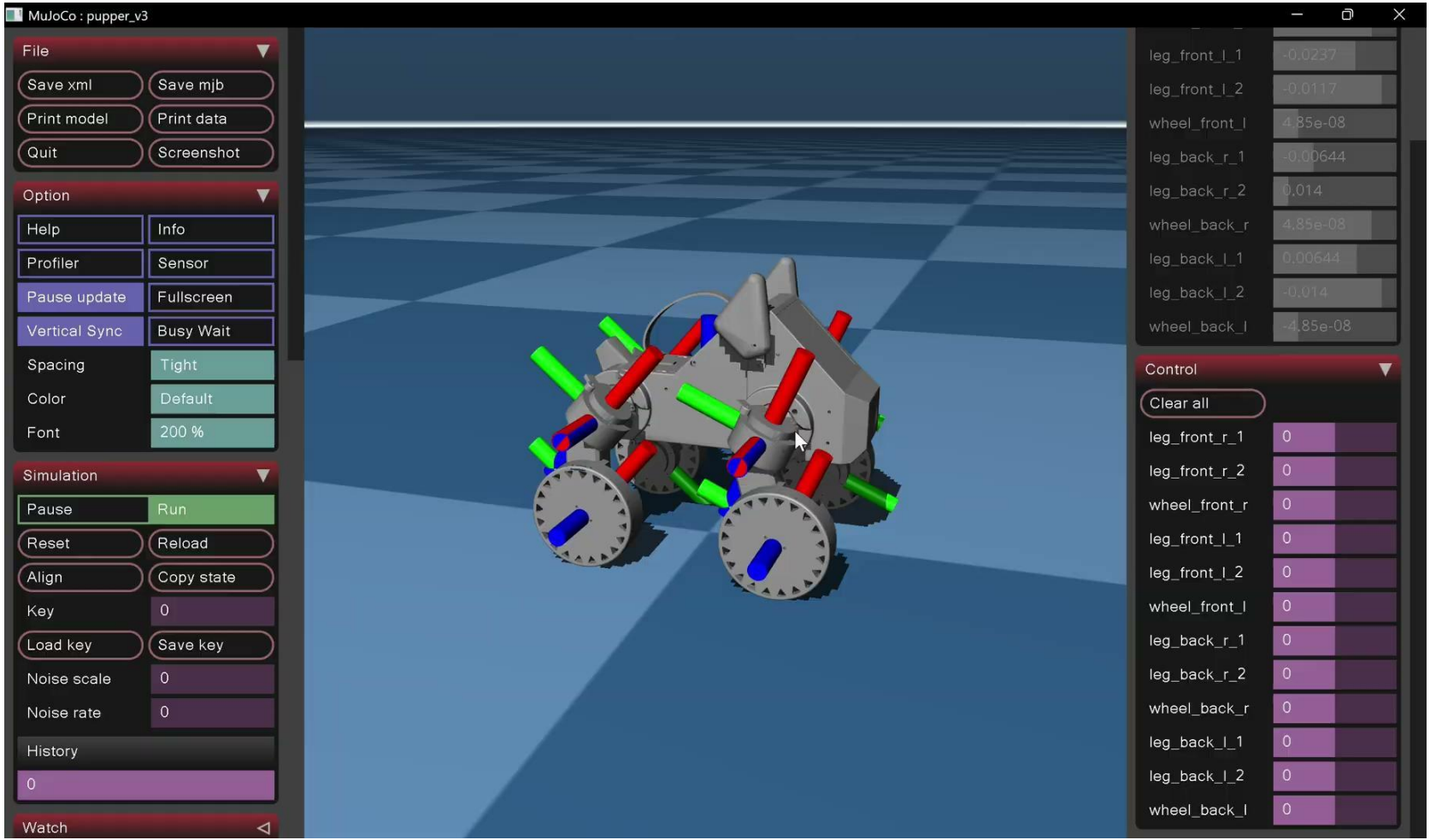
14/07/25 update

Progress

- Software 

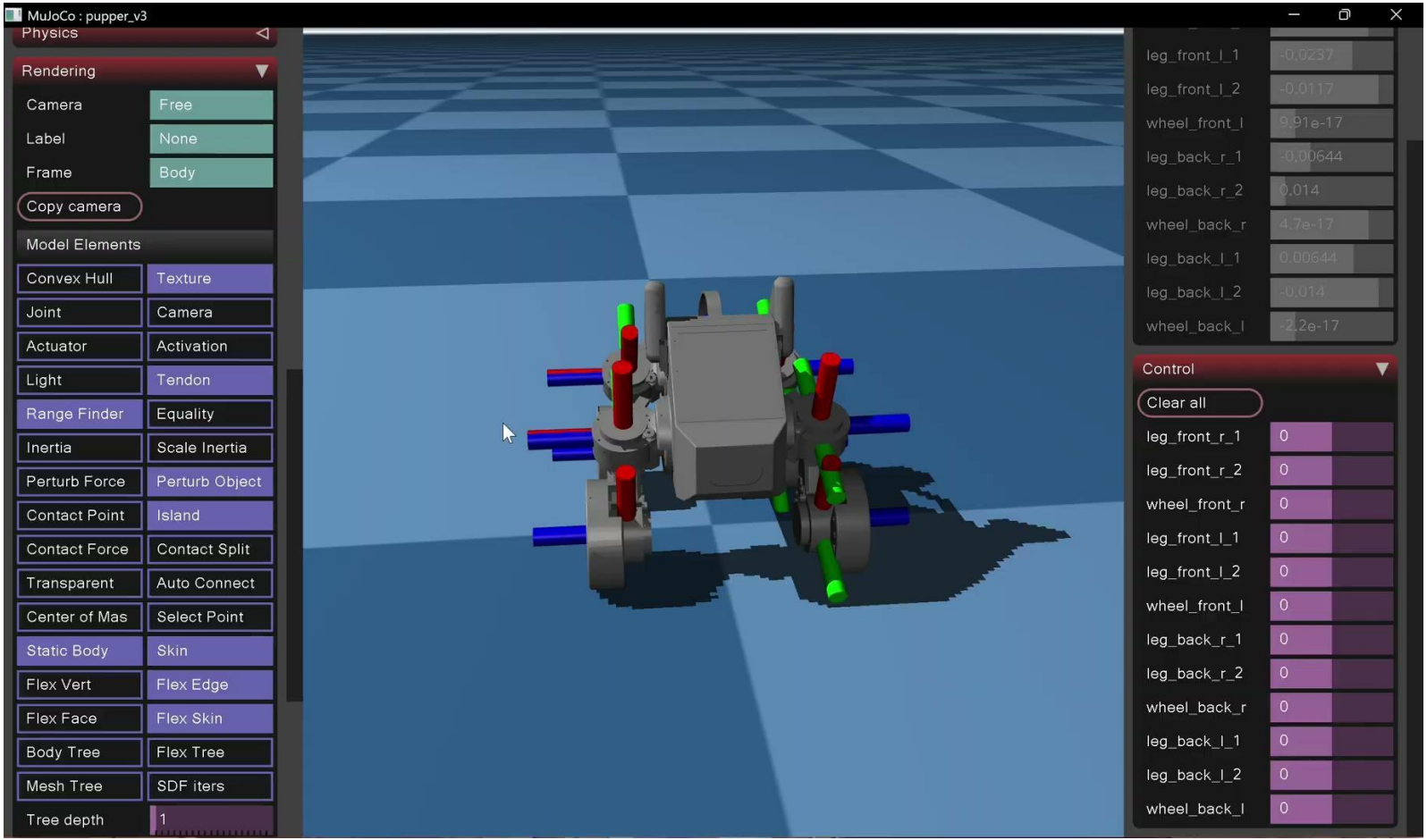
Wheel model - Tund

Software

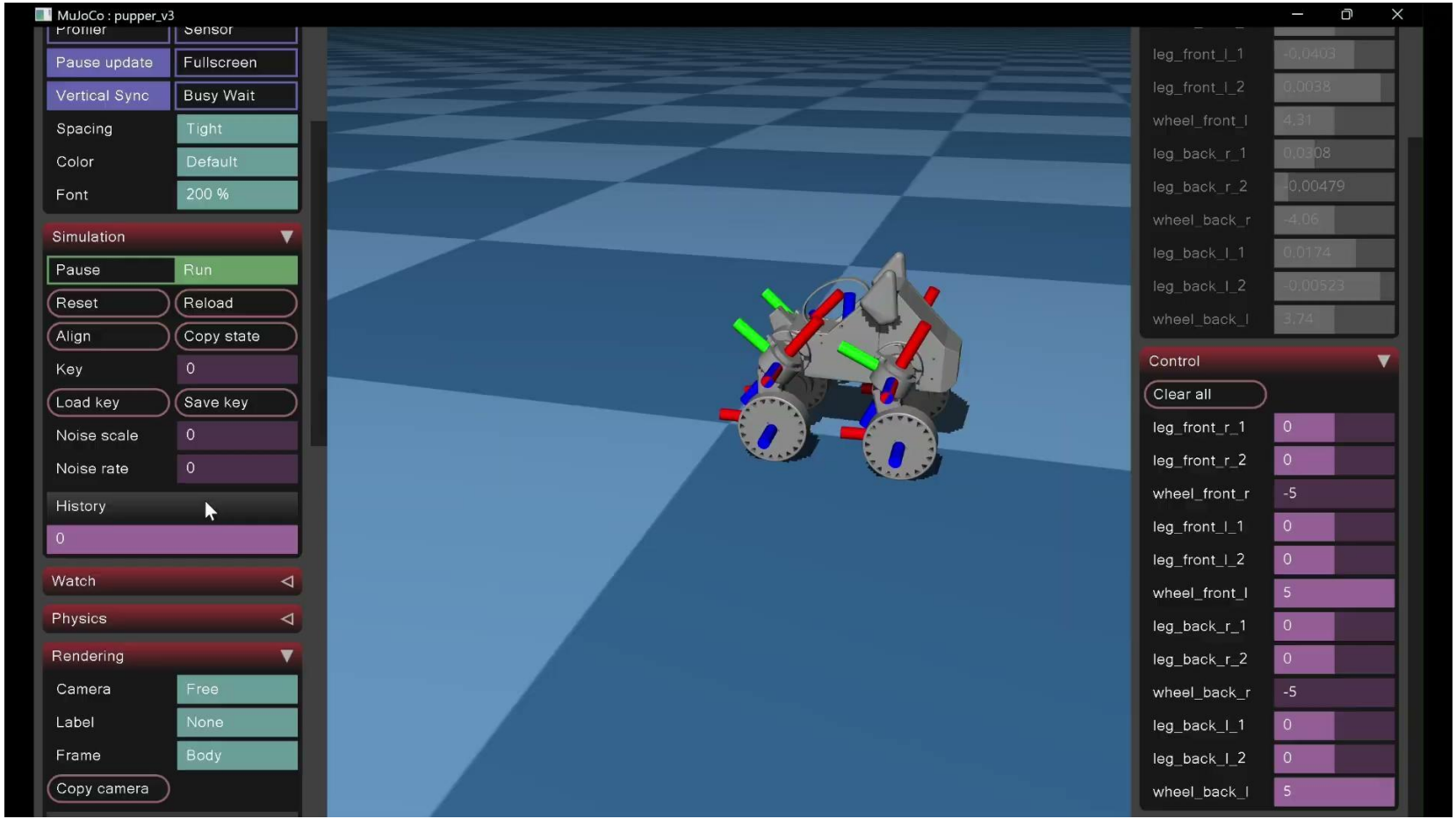


Wheel model - Moment of inertia

Software



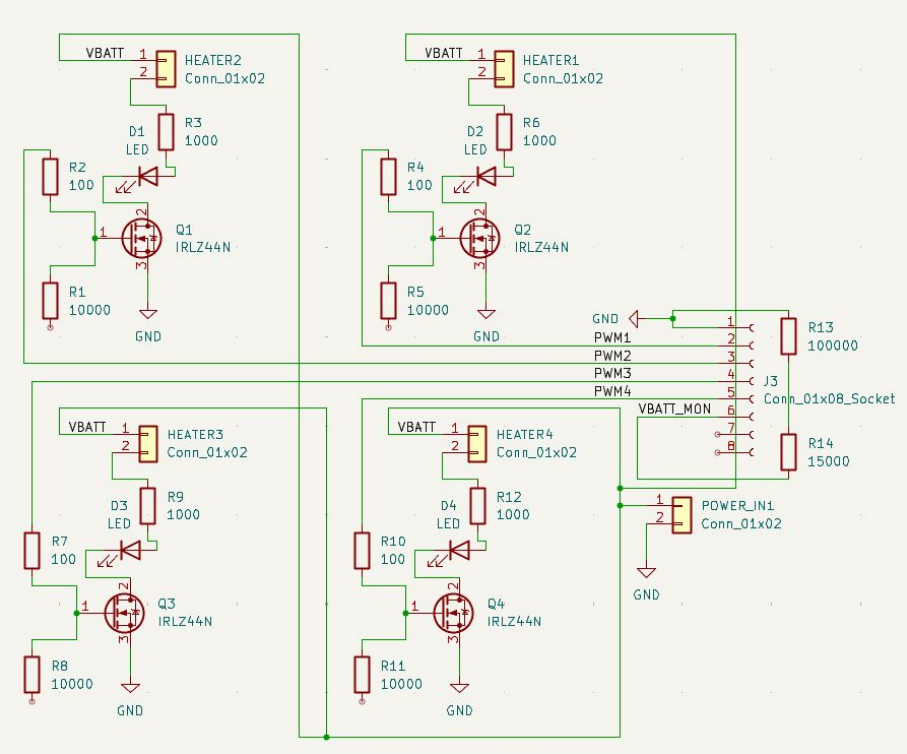
Wheel mode - Driving



Software

Software

PCB Schematic



- Heated by external VBATT
  - Mosfet switching
  - PWM control inputs from MCU
- Resistors
  - 10k ohm pull down resistors
  - 100 ohm limits inrush current and ringing
  - Voltage divider to check battery from MCU
- Diodes
  - Status LED’s with current limiting resistors.

Next Steps ⬆️

Next Steps ➡️ Hardware

- We’re still waiting on the new powerboards to arrive (2 week lead time)
- Prototype different wheels
  - Work with Dr. Kuangs team

Next Steps ➡️ Software

- Train wheel policy or develop control strategies for wheeled locomotion
- Continue transferring Colab RL pipeline to a local file

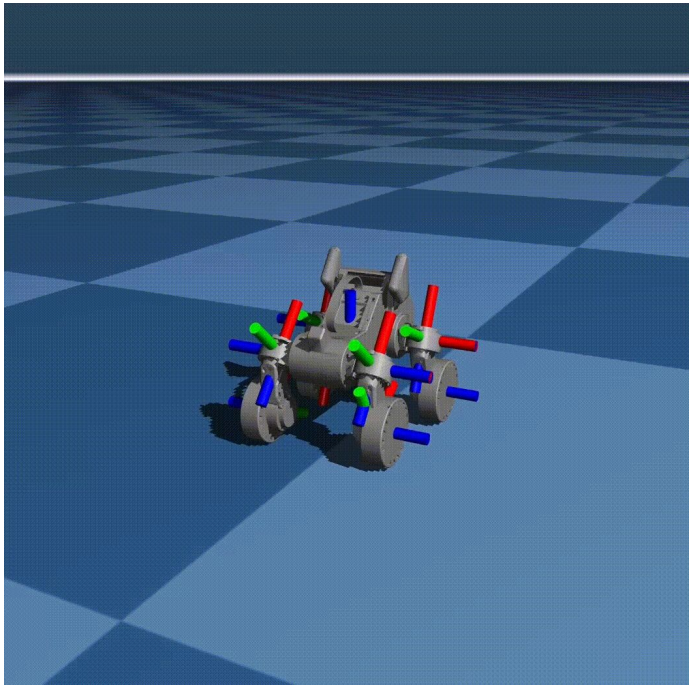
Pupper project

21/07/25 update

Software

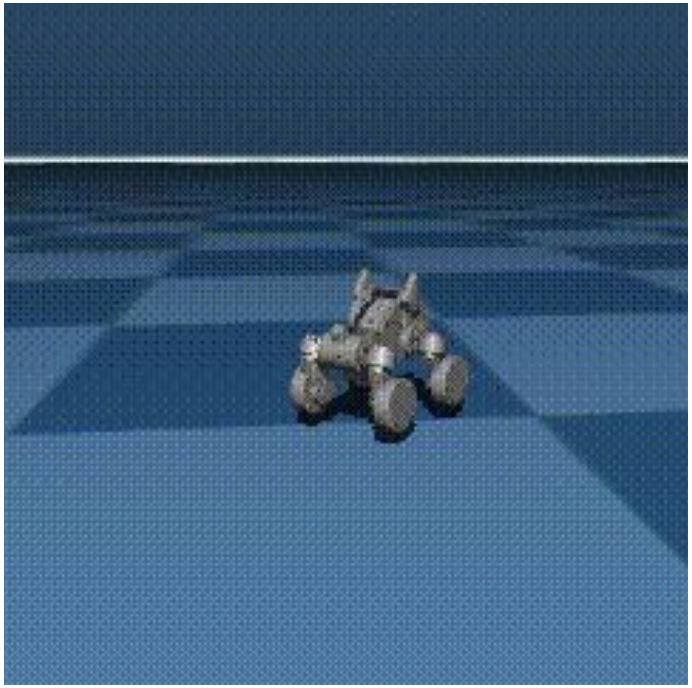
Continuing from last week ⏳

MJX 🤖



➡️

Import and get it to work with PPO policy



Software

Changing rewards for wheeled locomotion

🔵 Tracking Rewards (Increased)

Parameter	Old → New	Reason
tracking_lin_vel	1.5 → 2.5	Wheel robot better at maintaining lin_vel especially on flat terrain
tracking_ang_vel	0.8 → 1.5	Wheels enable precise yaw control through steering, increasing accurate turning when needed.
tracking_orientation	1 → 0	Wheeled robots only needs to stay level in Z, so full-body orientation tracking isn't necessary

🟠 Stability Penalties (Strengthened)

Parameter	Old → New	Reason
lin_vel_z	-2.0 → -5.0	Wheeled robots should not leave the ground—no jumping or bouncing.
ang_vel_xy	-0.05 → -2.0	Roll/pitch rates can indicate tipping. large penalty is applied to enforce flat motion.
orientation	-5.0 → -8.0	A stronger penalty for body tilt (roll/pitch angles) enforces ground-stable movement.

Software

Changing rewards for wheeled locomotion

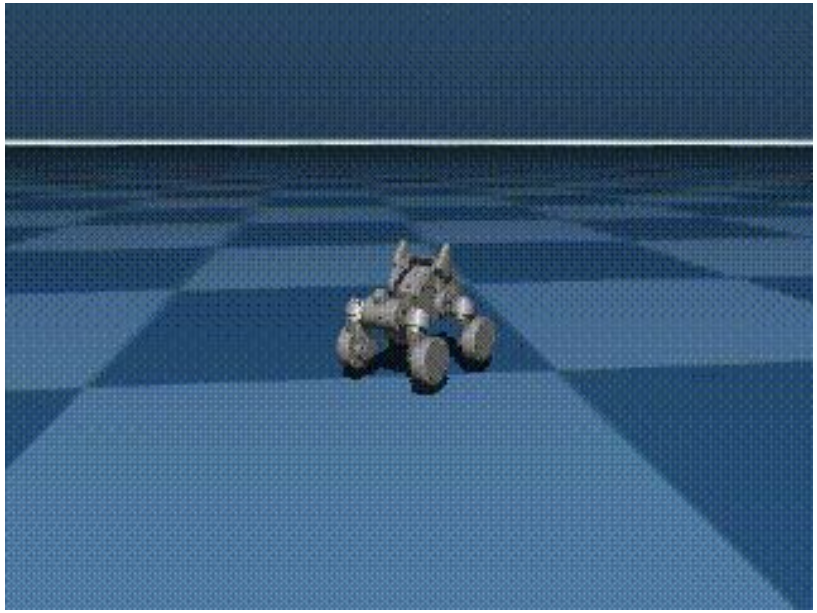
❌ Irrelevant Rewards for Wheeled Locomotion

Parameter	Old → New	Reason
feet_air_time	0.2 → 0	Encouraged leg swing in walking gaits—irrelevant for wheels. Disabled.
foot_slip	-0.1 → 0	Penalized foot sliding—not applicable since wheels are meant to roll.
knee_collision	-1.0 → 0	Legs/knees are not part of the robot—no longer applicable. Disabled.
abduction_angle	-0.1 → 0	Penalized leg splay—not relevant for wheeled structures. Disabled.
joint_acceleration	-1e-6 → 0	Penalized joint motion—wheels may not use joints. Disabled.
stand_still_joint_velocity	-0.1 → 0	Penalized idle joint motion—irrelevant with minimal joints. Disabled.

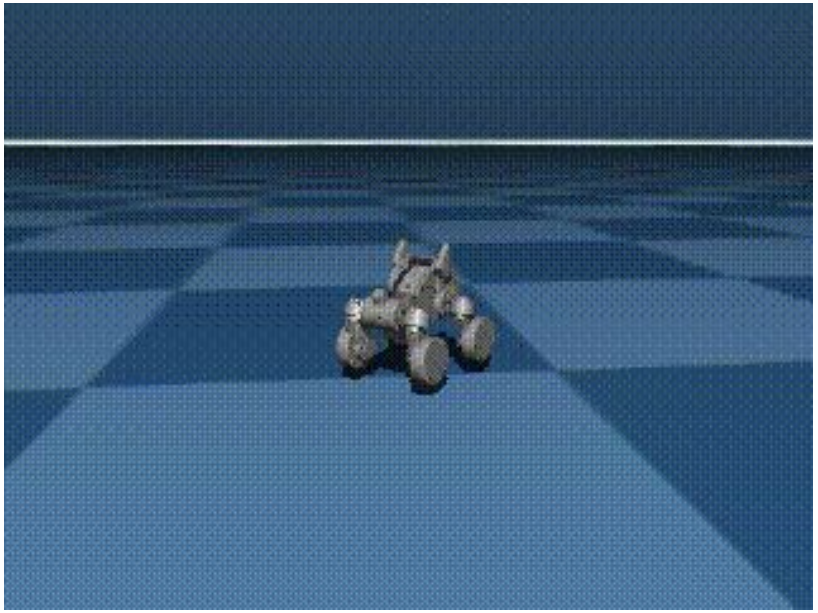
Software

Results

!?!? Why is it still hopping !?!?



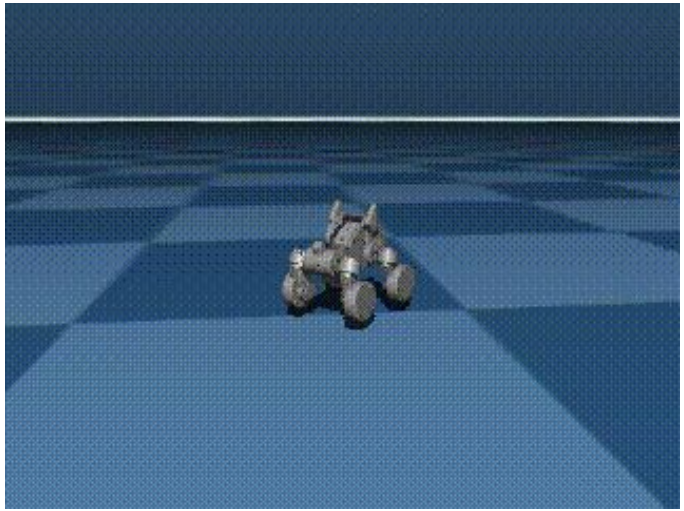
!?!? Sliding !?!?



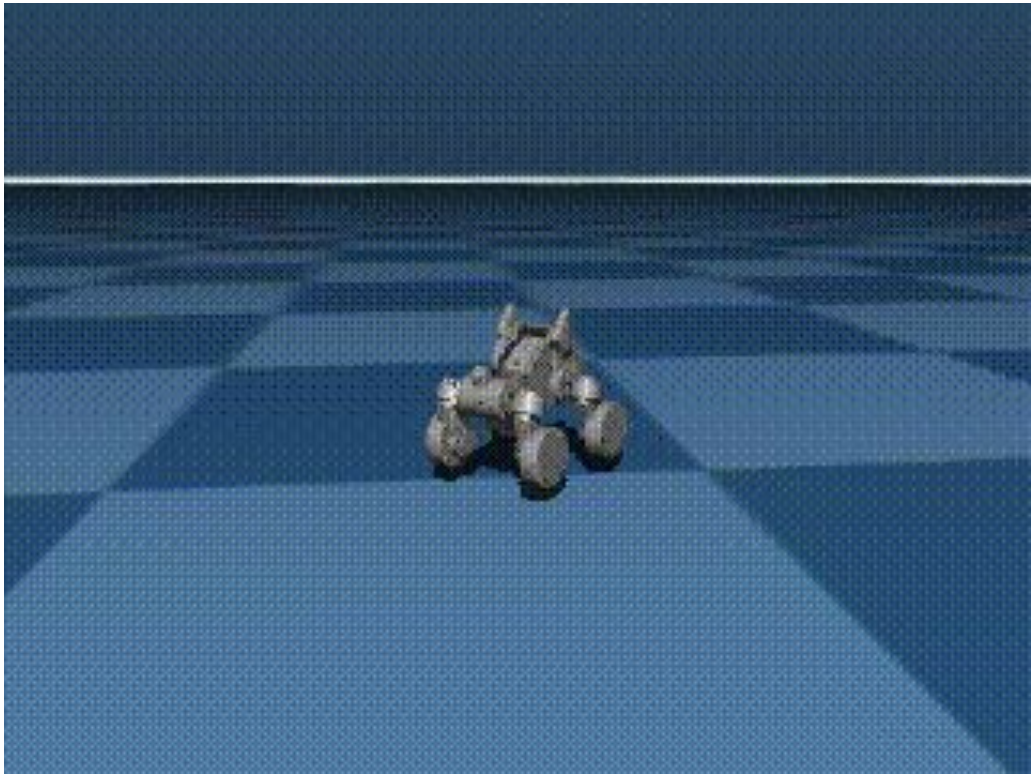
Software

Clues to why it's not working as intended 🤔

❌ Some sort of joint angle restriction ❌



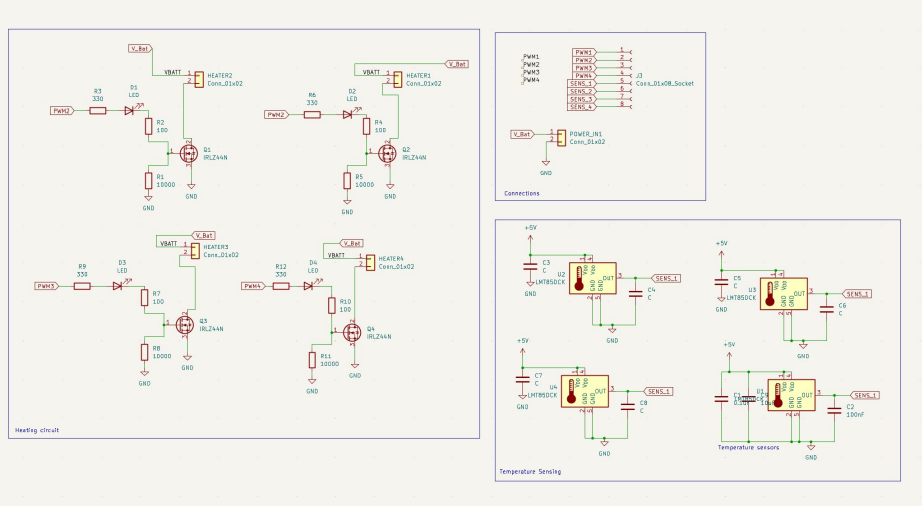
Body in the right orientation but wheels not moving



Wheels oscillating not rotating 360

Software

PCB Schematic



- PCB rev2
  - Temp. sensing
  - Cleaner schematic
  - Ordered test parts to make bread board circuit
- Determining Power Reqs.
  - P_batt = mc(dT)-P_air
    - Ambient temp may be negligible
    - P_batt could be modeled as I^2*R or (D*V^2)/R
      - R can be roughly approximated by element resistance

Next Steps ⬆️

Next Steps ➡️ Hardware

- Heat board parts should arrive in the next couple of days
- Assemble breadboard

Next Steps ➡️ Software

- Find and remove angle limit from the training policy and get a working wheeled policy by next week

Weekly Progress - Working Pupper (Mostly)

Ethan Dretzka

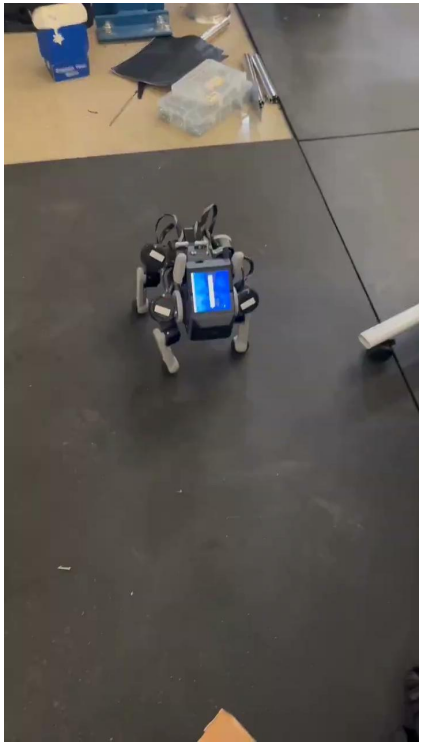
Tund Theerawit

Pat Zhu

Matthew Suri

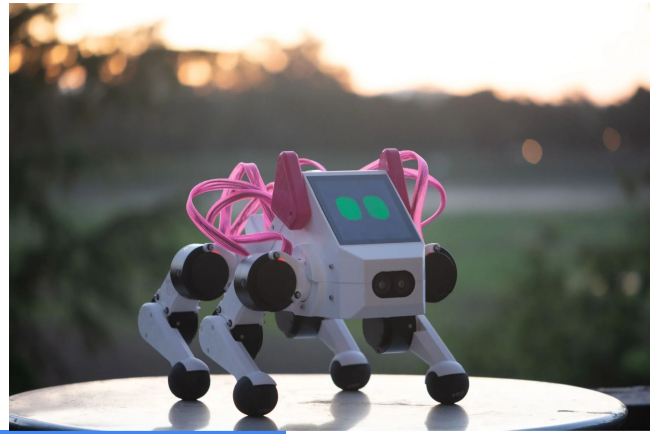
Fully Assembled

Walking with PS5 Controller



Next Steps

- Source new motor control board or motor
  - Current board is burned out
- Upload simulation based policy
- Validate trainings
- Meet with Professor Kuang



Pupper project

23/09/25 update

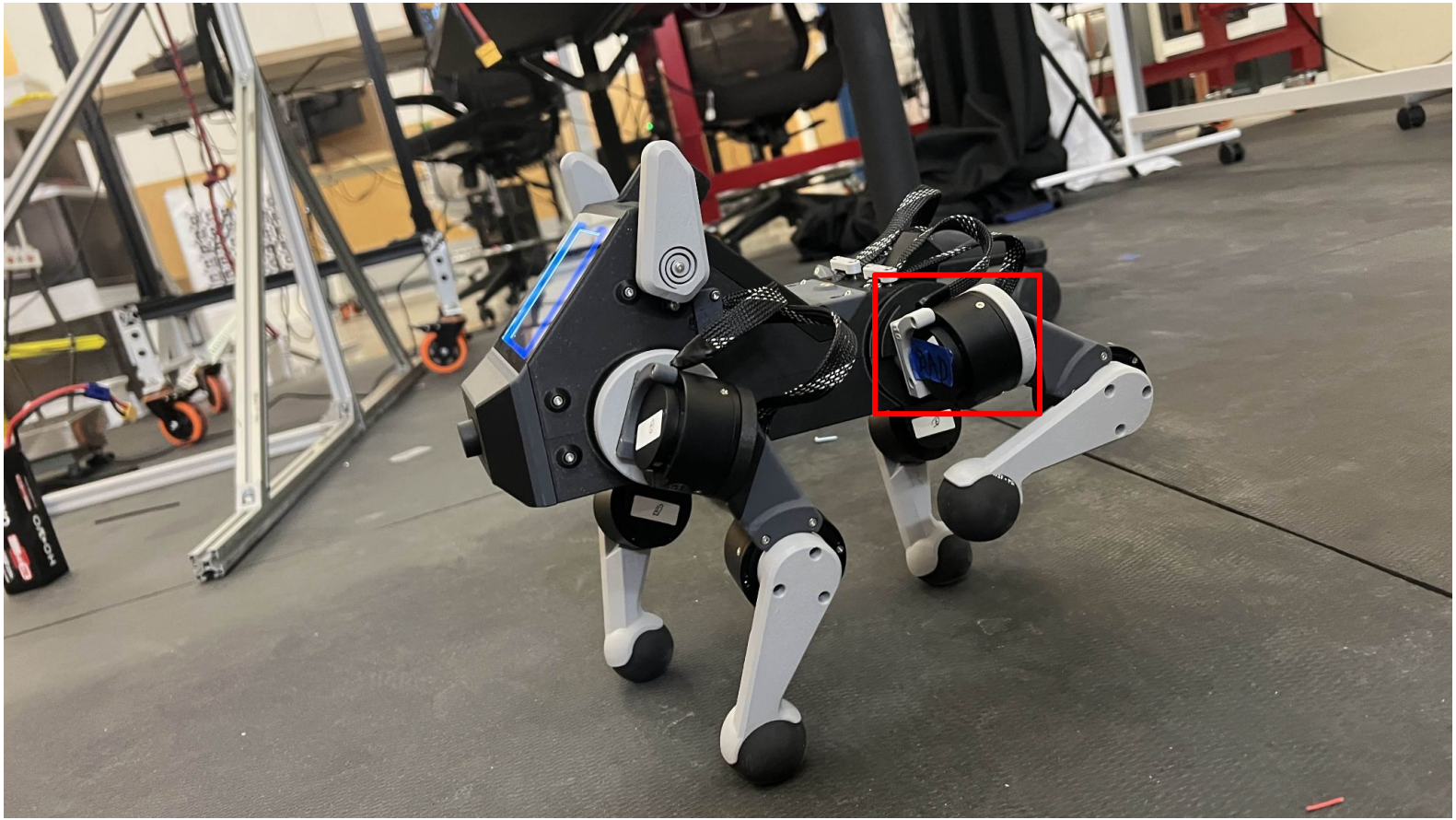
Tund, Ethan, Pat, Matt

Progress

- Software 
- Hardware 

Hardware

Waiting on motor 🕒



Hardware

3 Leg policy robustness testing 🔭

Software 

Software

Fix training pipeline

Jax and Orbax version misalignment

🔍 What the PyPI / GitHub info says

- The **latest release** of `orbax-checkpoint` on PyPI is **version 0.11.25**, released **September 11, 2025.** PyPI +1
- A version just before that, **0.11.24**, came out August 28, 2025. PyPI +1
- In the release notes for 0.11.25, one of the notable changes is: *"Bump minimum JAX version to v0.6.0"*.

GitHub

```
[1] ---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
/tmp/ipython-input-1734472816.py in <cell line: 0>()
     39
     40 # Joint limits
---> 41 sys_temp = mjcf.load(simulation_config.original_model_path)
     42 joint_upper_limits = sys_temp.jnt_range[1:, 1]
     43 joint_lower_limits = sys_temp.jnt_range[1:, 0]

                                 29 frames
/usr/local/lib/python3.12/dist-packages/jaxlib/triton/_triton_ops_gen.py in <module>
     12 )
     13 _ods_ir = _ods_cext.ir
---> 14 _ods_cext.globals.register_traceback_file_exclusion(__file__)
     15
     16 import builtins

AttributeError: 'jaxlib.mlir._mlir_libs._mlir._Globals' object has no attribute 'register_traceback_file_exclusion'
```

Software

Fix training pipeline

Possible fixes - directly specify compatible versions

OLD

```
!pip install -q mujoco==3.2.7 mujoco-mjx==3.2.7 brax==0.10.5 flax==0.10.2 orbax==0.1.9
!pip install black[jupyter] --quiet
import os
# Tell XLA to use Triton GEMM, this improves steps/sec by ~30% on some GPUs
xla_flags = os.environ.get('XLA_FLAGS', '')
xla_flags += ' --xla_gpu_triton_gemm_any=True'
os.environ['XLA_FLAGS'] = xla_flags
# Clean up incompatible jax versions
!pip uninstall -y jax jaxlib #orbax-checkpoint

# Install specific compatible jax versions
!pip install jax==0.5.0 jaxlib==0.5.0 -f https://storage.googleapis.com/jax-releases/jax_cuda_releases.html
```

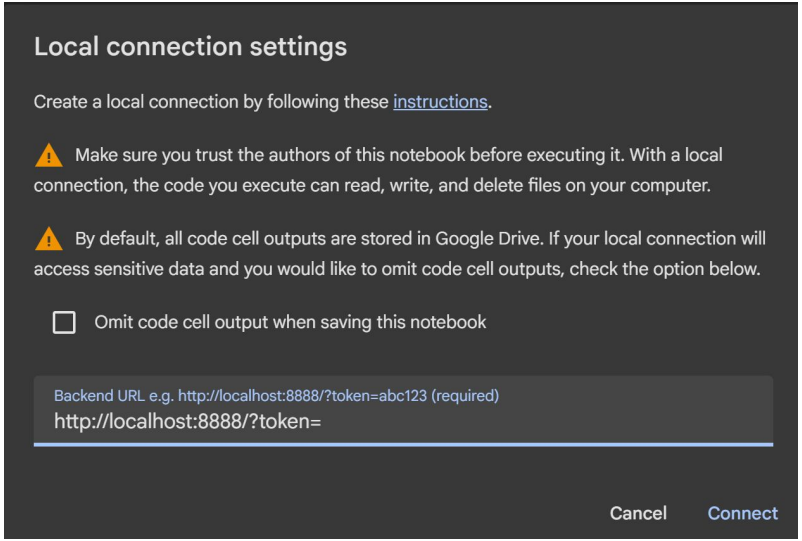
NEW ⭐

```
!pip install -q mujoco==3.2.7 mujoco-mjx==3.2.7 brax==0.10.5 flax==0.10.2 orbax==0.1.9
!pip install black[jupyter] --quiet
import os
# Tell XLA to use Triton GEMM, this improves steps/sec by ~30% on some GPUs
xla_flags = os.environ.get('XLA_FLAGS', '')
xla_flags += ' --xla_gpu_triton_gemm_any=True'
os.environ['XLA_FLAGS'] = xla_flags
# Clean up incompatible jax versions
!pip uninstall -y jax jaxlib #orbax-checkpoint

# Install specific compatible jax versions
pip install jax==0.5.0 jaxlib==0.5.0 orbax-checkpoint==0.6.0 -f https://storage.googleapis.com/jax-releases/jax_cuda_releases.html
```

Software

Colab egress

- Ask Hanwen if the 5090 PC is backed up.
- If it is, try running our policy there instead of loading up Colab each time.
- While waiting for components to arrive, use the time as a learning opportunity:
- Write our own in-house PPO policy (based on the Stanford one).
- Run it locally on the lab PC.



Pupper project

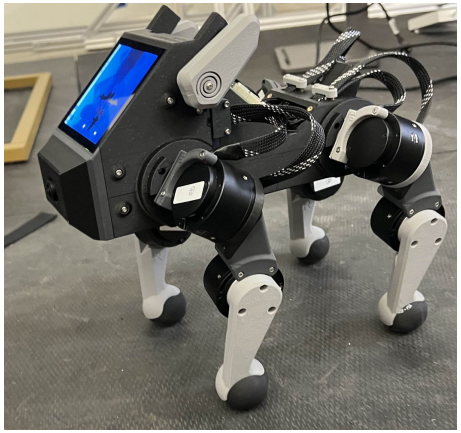
30/09/25 update

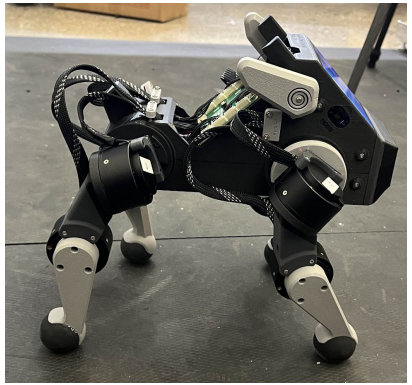
Tund, Ethan, Pat, Matt

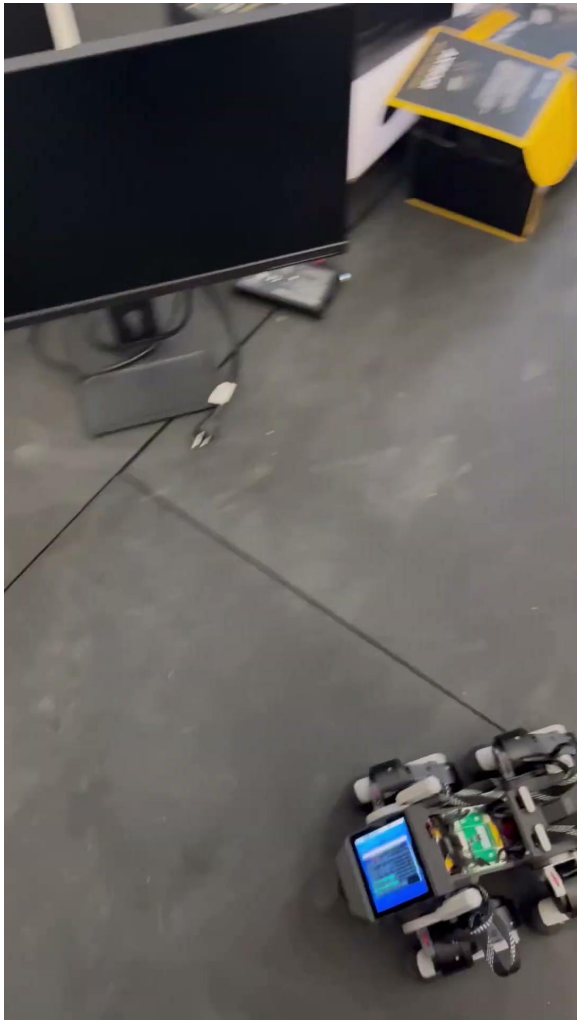
Hardware 🔧

Hardware

Motors are here!!! 😮



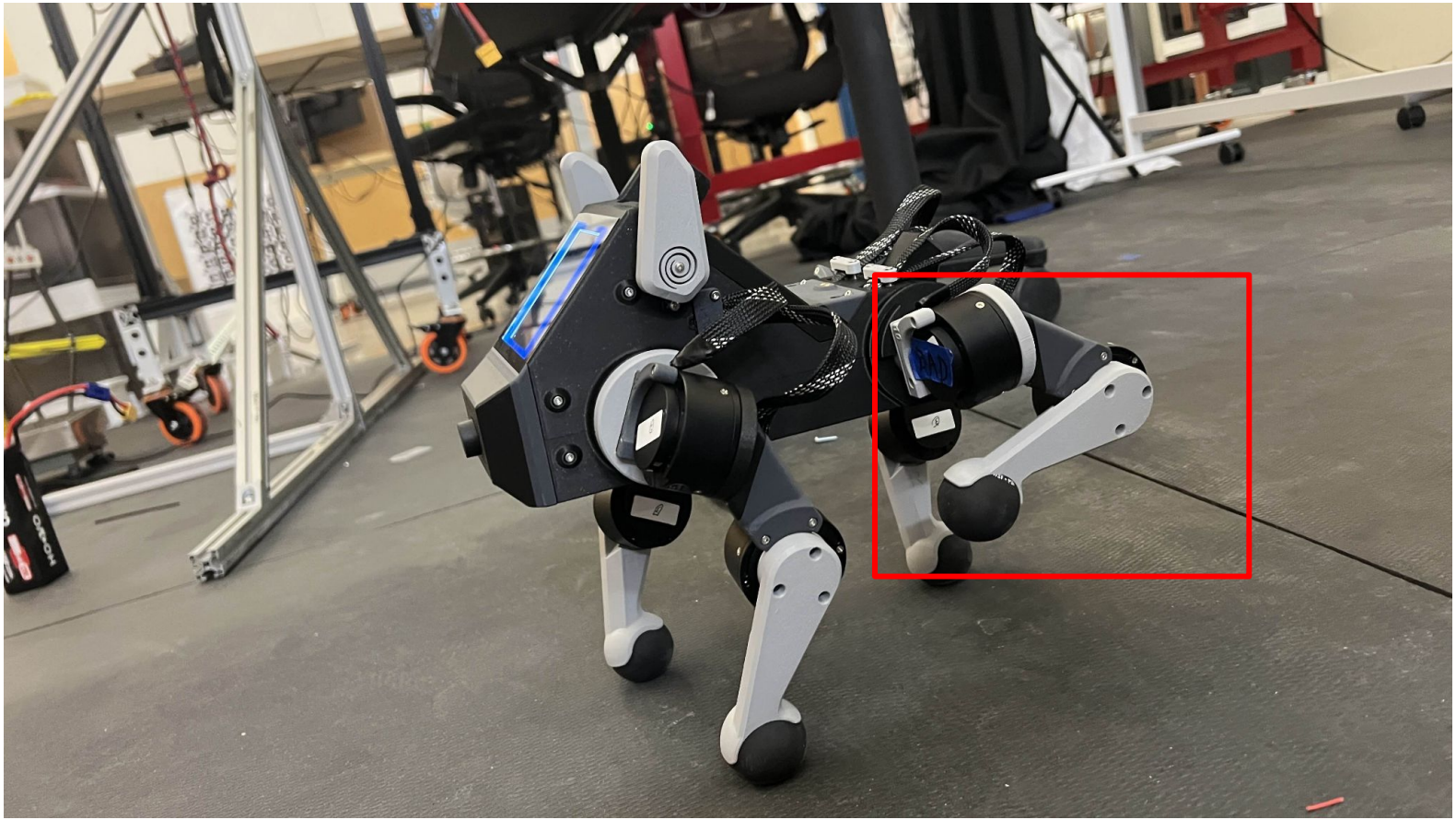




Software 

Hardware

Expecting counter weight 🕒

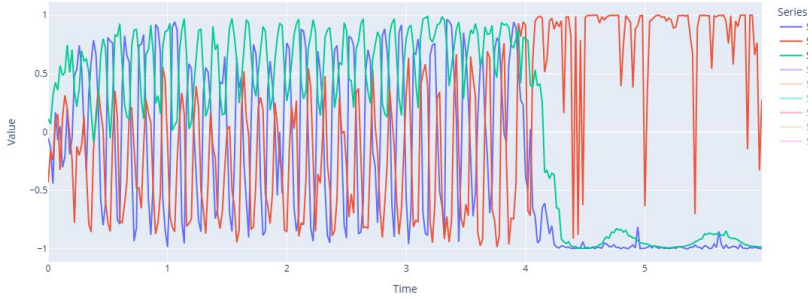


Software

Exploring 3 legged policy

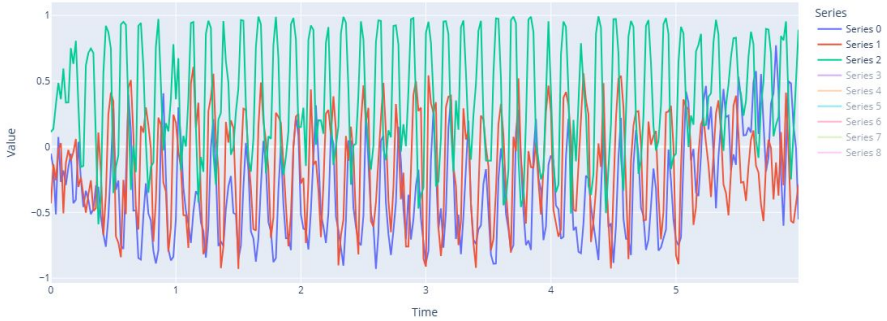
Random movement

Policy output



Forward movement

Policy output

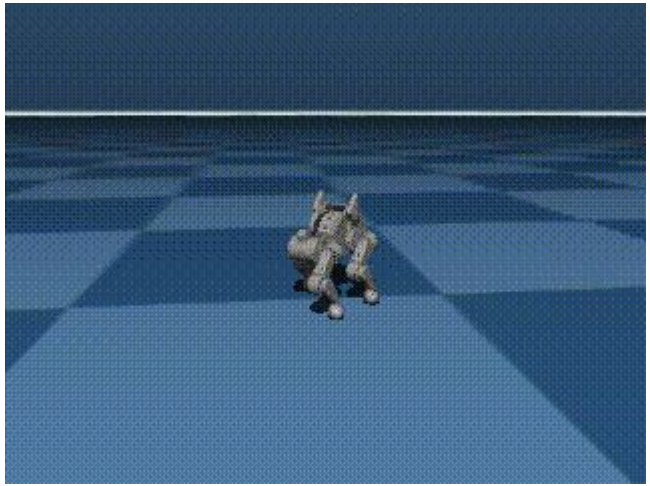


Software

Exploring 3 legged policy 🦿🦿🦿

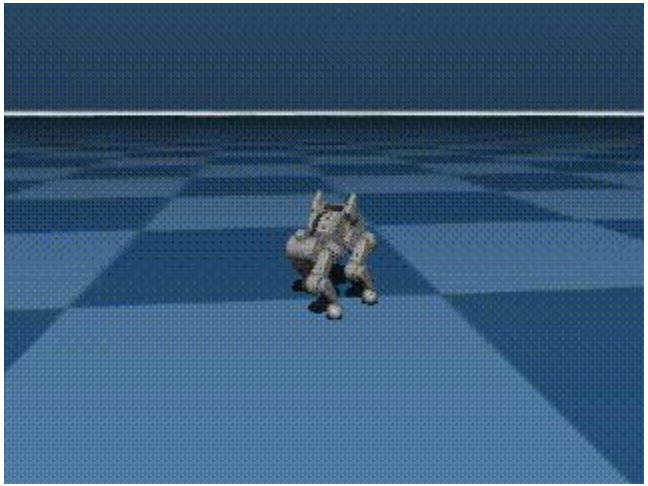
Lessons learned

Old
Leg getting stuck moving backwards



➡️

New
What changed ⁉️

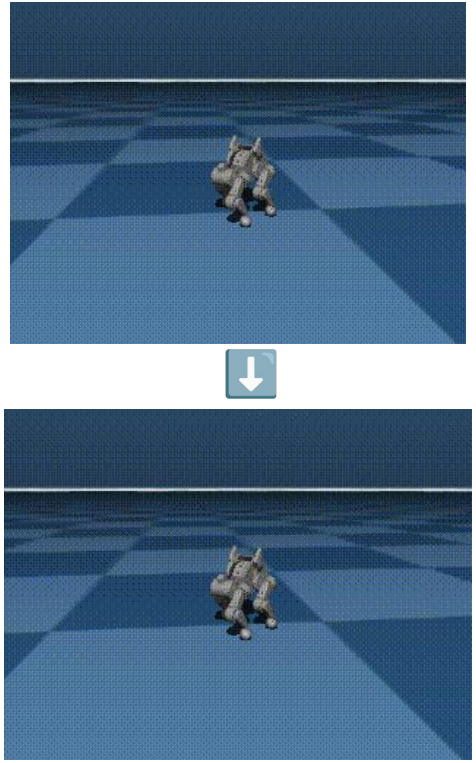


Software

Exploring 3 legged policy 🦿🦿🦿

Lessons learned

Leg getting stuck moving backwards



What changed ⁉️

1) ⏳Trained for longer

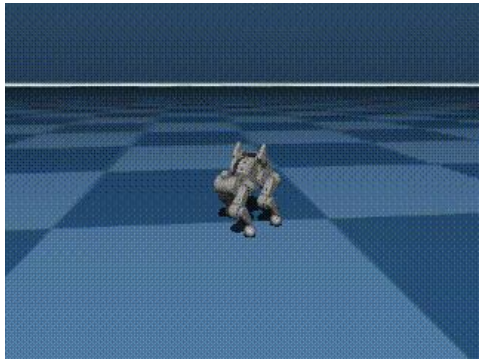
2) 🧠Edit reward weights

Software

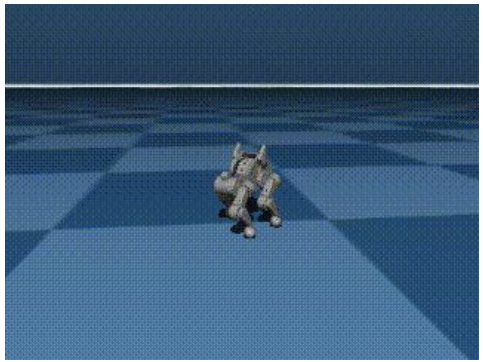
Exploring 3 legged policy

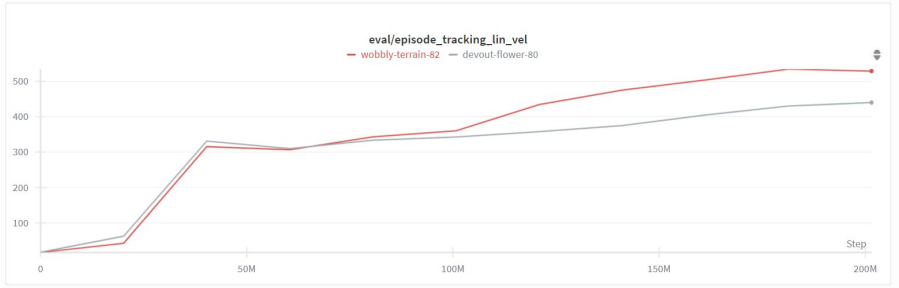
Lessons learned

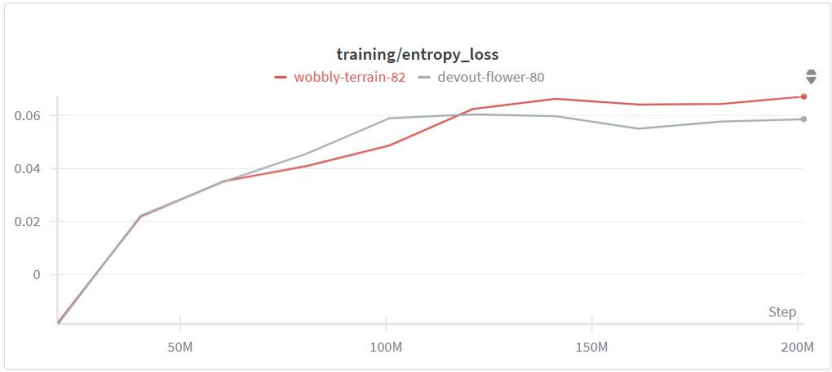
devout-flower-80 (Leg stuck)



wobbly-terrain-82





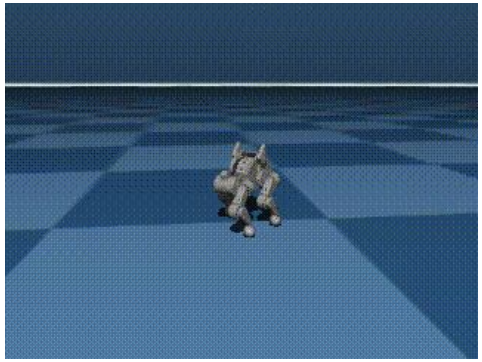


Software

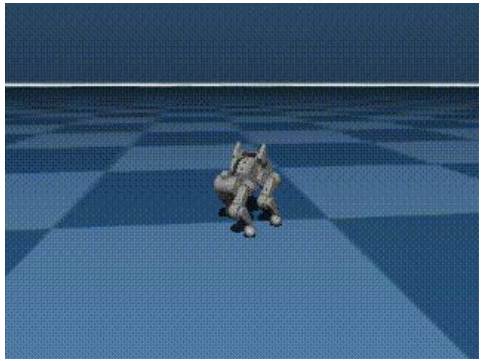
Exploring 3 legged policy

Lessons learned

devout-flower-80 (Leg stuck)



wobbly-terrain-82



Changes

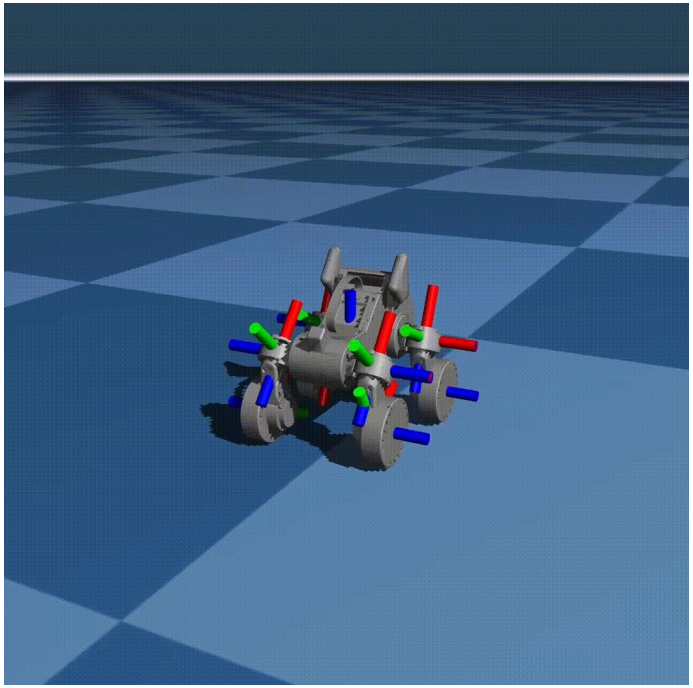
```
35
36 # Penalize the change in the action and encourage smooth
37 # actions. L1 regularization |action - last_action|^2
38-reward_config.rewards.scales.action_rate = -0.01
39
40 # Encourage long swing steps. However, it does not
41 # encourage high clearances.
42-reward_config.rewards.scales.feet_air_time = 0.2
43
44 # Encourage joints at default position at zero command, L1 regularization
45 # |q - q_default|.
46 reward_config.rewards.scales.stand_still = -0.5
47
48 # Encourage zero joint velocity at zero command, L1 regularization
49 # |q_dot|.
50 # Activates when norm(command) < stand_still_command_threshold
51 # Commands below this threshold are sampled with probability zero_command_probability
52 reward_config.rewards.scales.stand_still_joint_velocity = -0.1
53
54 # Encourage zero abduction angle so legs don't spread so far out
55 # L2 loss on ||abduction_motors - desired||^2
56-reward_config.rewards.scales.abduction_angle = -0.1
57
```

```
35
36 # Penalize the change in the action and encourage smooth
37 # actions. L1 regularization |action - last_action|^2
38+reward_config.rewards.scales.action_rate = -0.02
39
40 # Encourage long swing steps. However, it does not
41 # encourage high clearances.
42+reward_config.rewards.scales.feet_air_time = 0.25
43
44 # Encourage joints at default position at zero command, L1 regularization
45 # |q - q_default|.
46 reward_config.rewards.scales.stand_still = -0.5
47
48 # Encourage zero joint velocity at zero command, L1 regularization
49 # |q_dot|.
50 # Activates when norm(command) < stand_still_command_threshold
51 # Commands below this threshold are sampled with probability zero_command_probability
52 reward_config.rewards.scales.stand_still_joint_velocity = -0.1
53
54 # Encourage zero abduction angle so legs don't spread so far out
55 # L2 loss on ||abduction_motors - desired||^2
56+reward_config.rewards.scales.abduction_angle = 0
57
```

Software

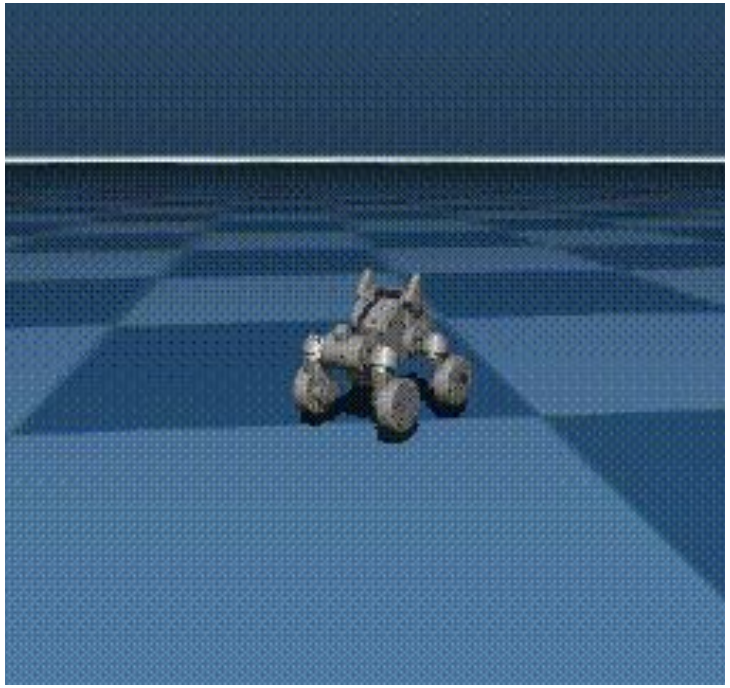
Continuing from last week ⏳

MJX 🤖



➡️

I know what's wrong and can fix with in the day



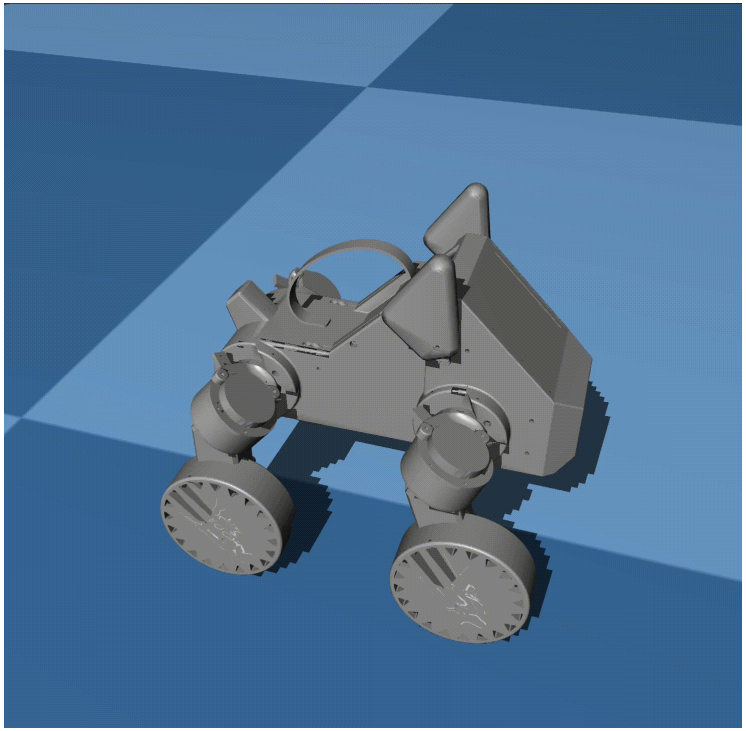
Pupper project

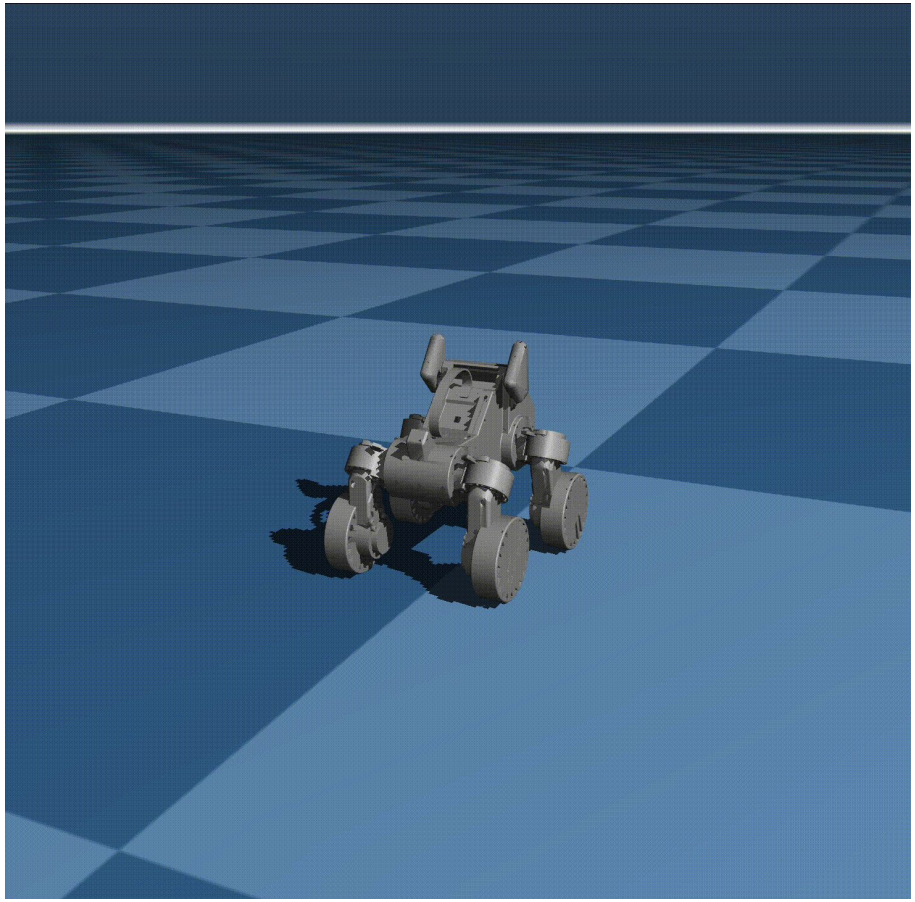
07/10/25 update

Tund, Ethan, Pat, Matt

Software

Wheel policy 🦽

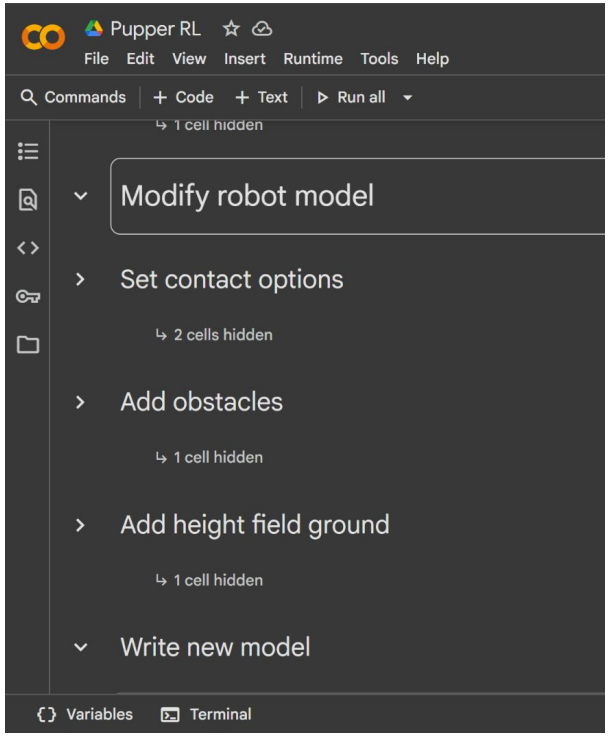




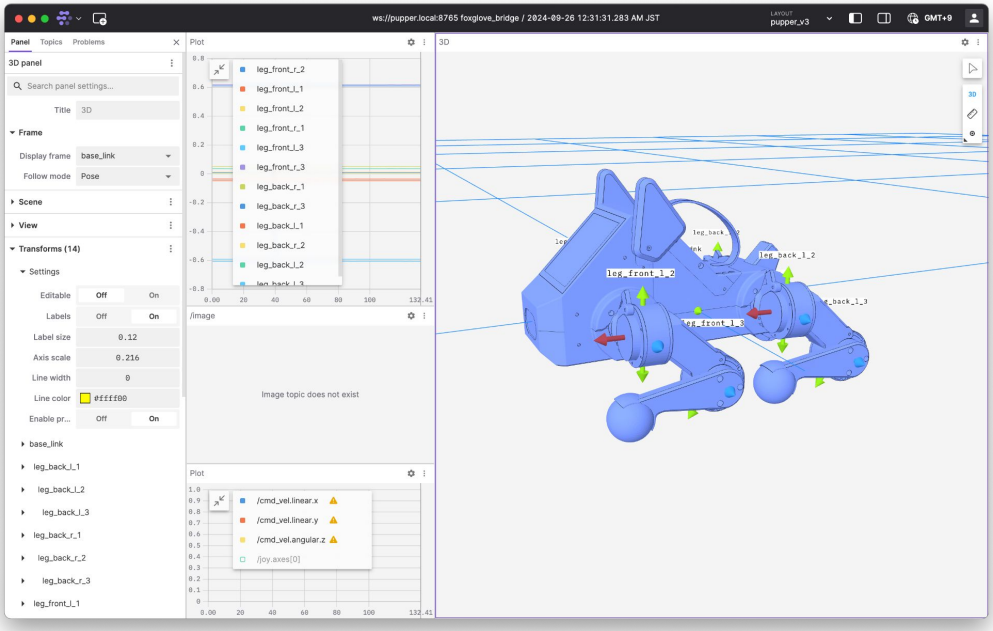
Software

Housekeeping 🧹

Simplified Colab



Foxglove - Visualization



Hardware

Hardware delays 🕑

Reprinting part

Resizable limbs

Next Steps ⬆️

Next Steps ➡️ Hardware

- Resizable limbs
- Run policy on real robot
  - Wheel
  - 3 leg

Next Steps ➡️ Software

- Get foxglove working
- Continue working on local training

Pupper project

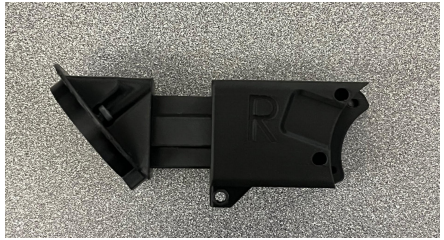
06/11/25 update

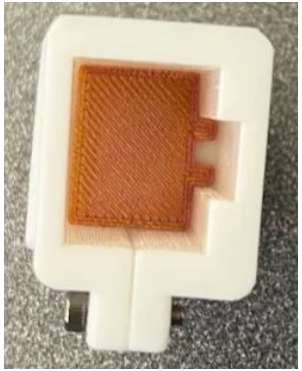
Tund, Ethan, Pat, Matt

Hardware 🔧

Hardware

Resizable legs & Test stand

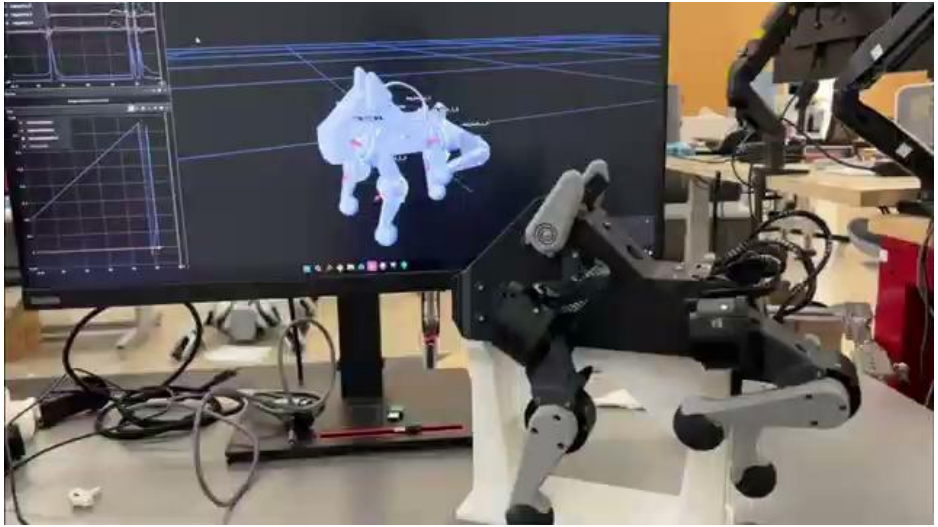


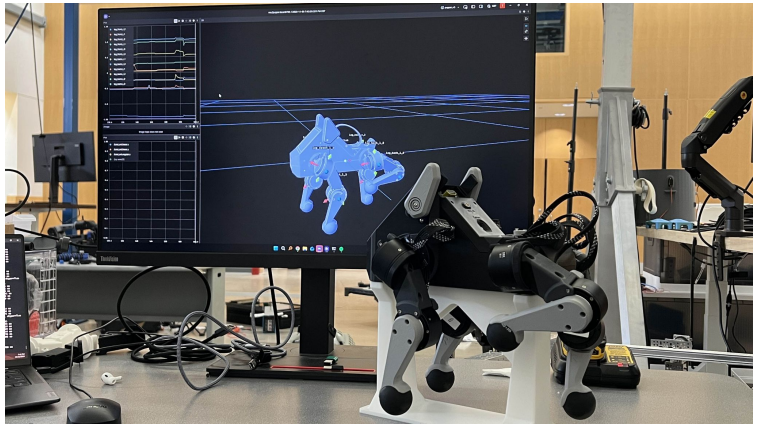


Software 

Software

Foxglove live visualisation 🦊🦊





Next Steps ⬆️

Next Steps ➡️ Hardware

- Waiting on Prof Kuang's lab

Next Steps ➡️ Software

- More RL knowledge needed working on stanford course would be beneficial

Lecture	Lab
Lecture 1: ROS Introduction and PD Control	Lab 1: ROS Introduction and PD Control
Lecture 2: Forward Kinematics	Lab 2: Forward Kinematics
Lecture 3: Inverse Kinematics	Lab 3: Inverse Kinematics and Trajectory Tracking
Lecture 4: Heuristical Gait Control	Lab 4: Model-Based Control and Trotting Gait Implementation
Lecture 5: Reinforcement Learning for Robotics	Lab 5: How to Train Your Dog
Lecture 6: Large Language Models for Robotics	Lab 6: Do What I Say
Lecture 7: Computer Vision for Robotics	Lab 7: The World I See

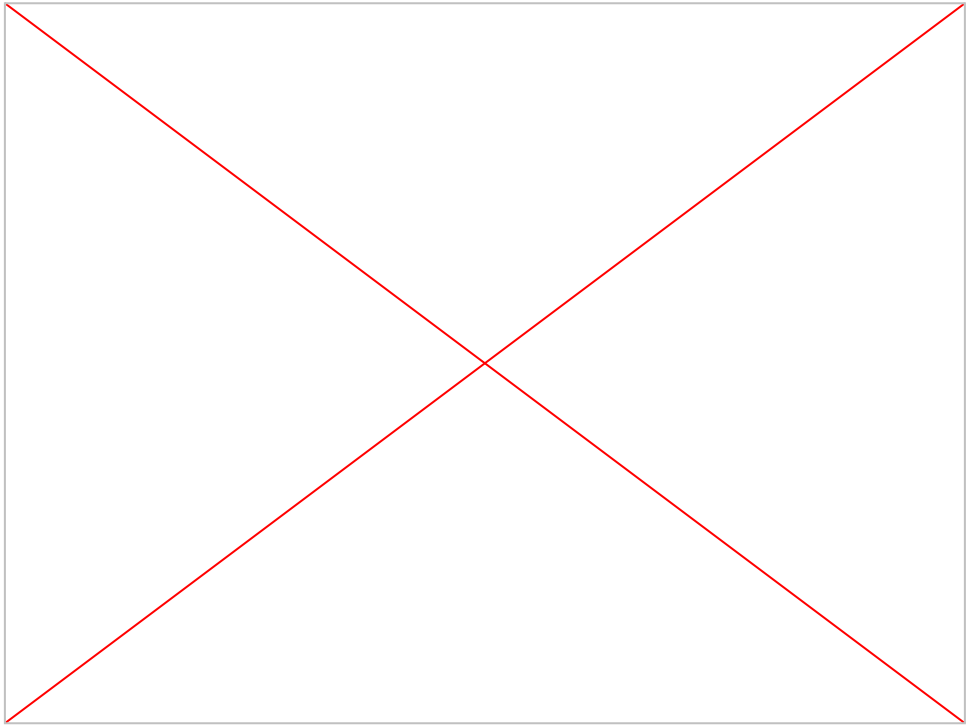
Pupper project

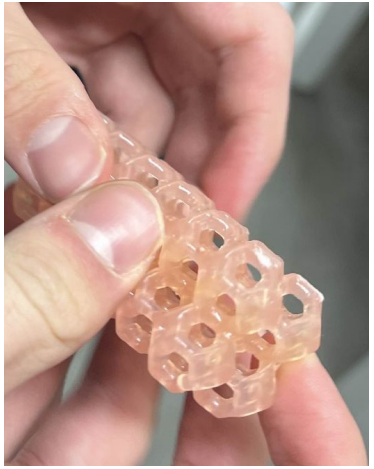
13/11/25 update
Tund

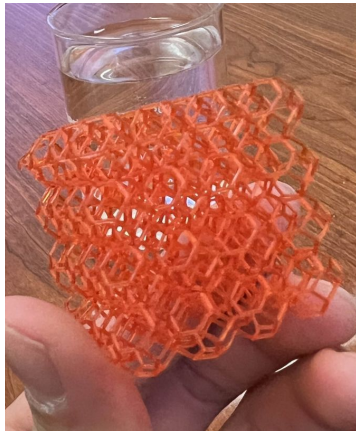
Hardware 🔧

Hardware

Shape memory material





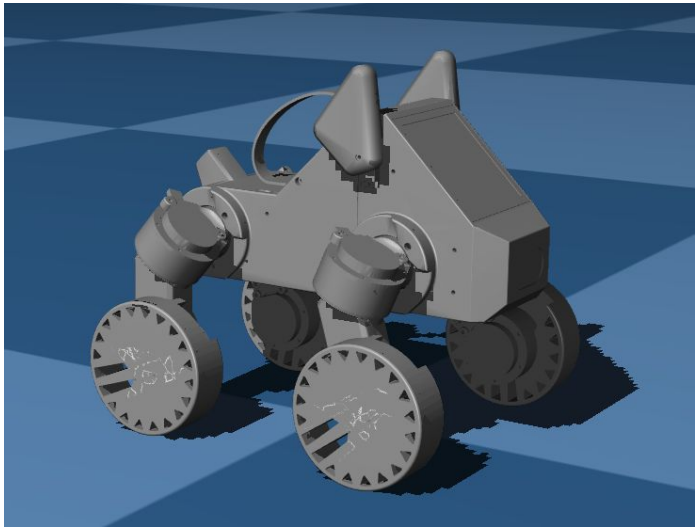


Software 

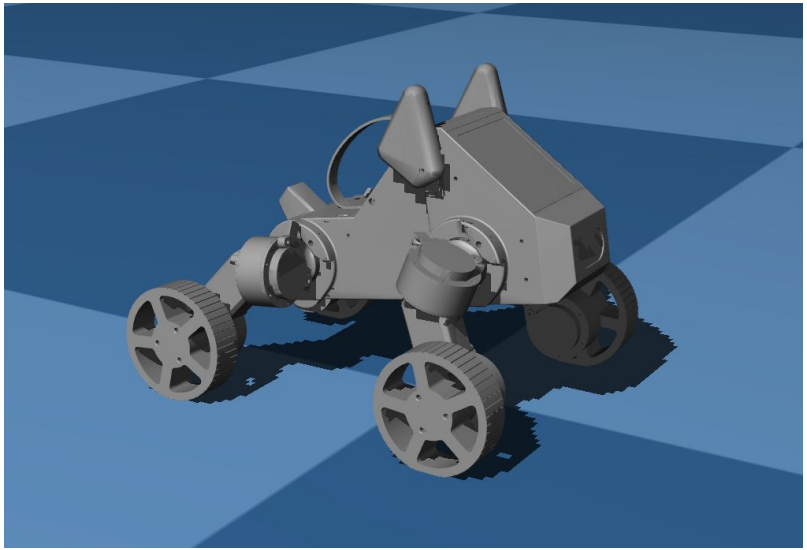
Software

Wheel Policy debug

Old



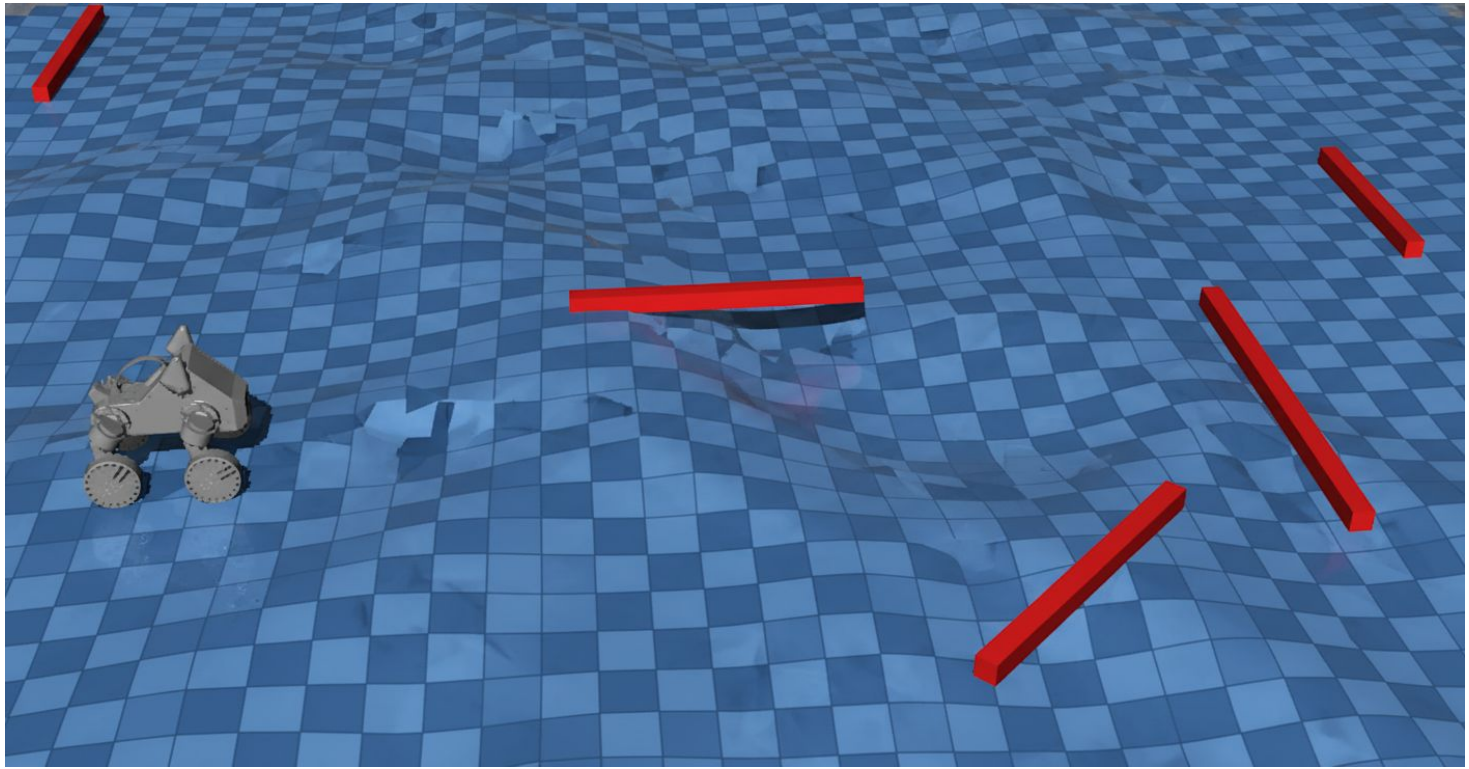
New



Software

Wheel Policy debug

Obstacles added - still needs to be ran through rl policy



Software

Wheel Policy debug

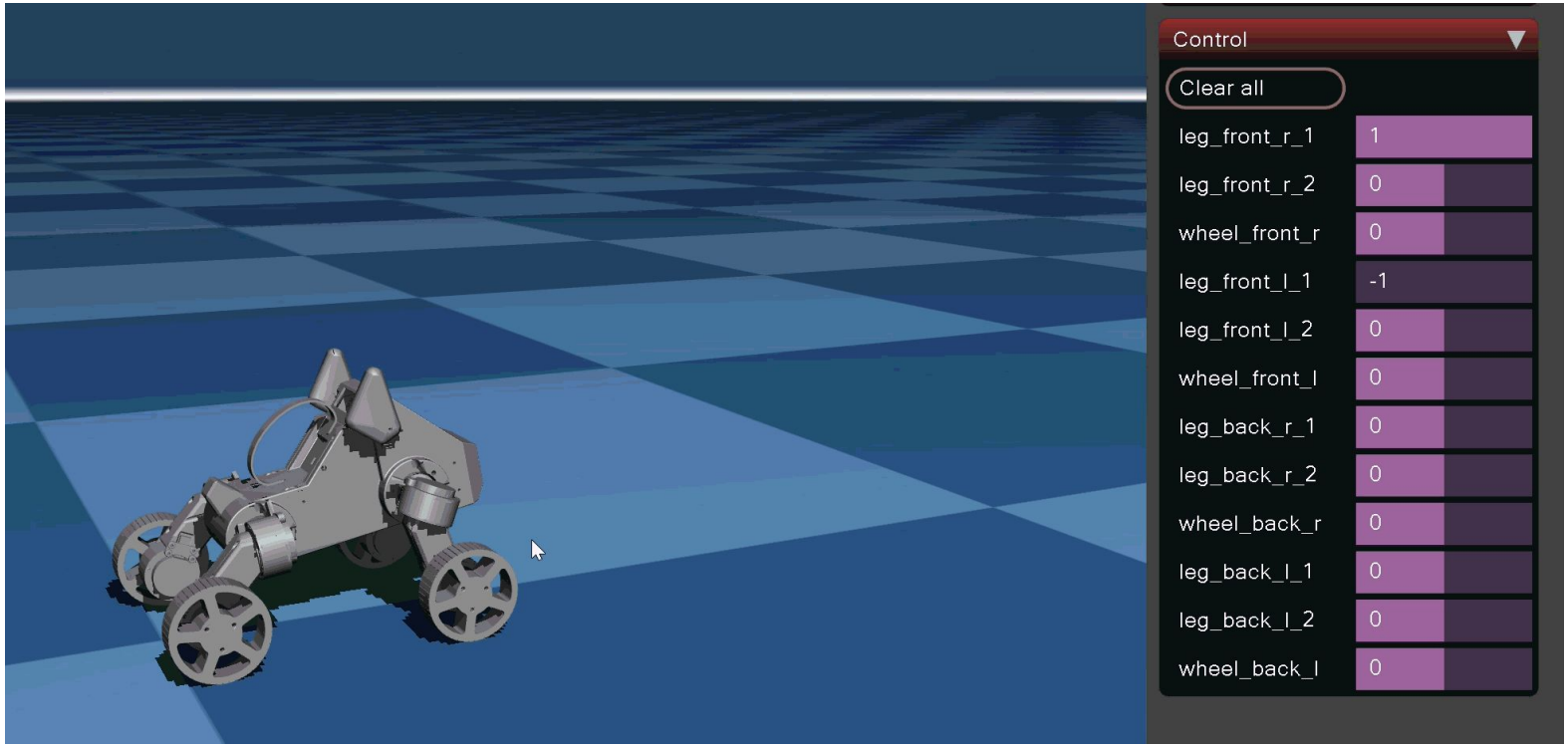
Changing from joint angle control to velocity control

```
<!-- ===================================================================================== -->
<!-- |                                    Actuators                                      | -->
<!-- ===================================================================================== -->
<actuator>
    <!-- Front Right Leg Actuators -->
    <general joint="leg_front_r_1" name="leg_front_r_1" />
    <general joint="leg_front_r_2" name="leg_front_r_2" />
    <velocity joint="wheel_front_r" name="wheel_front_r" />

    <!-- Front Left Leg Actuators -->
    <general joint="leg_front_l_1" name="leg_front_l_1" />
    <general joint="leg_front_l_2" name="leg_front_l_2" />
    <velocity joint="wheel_front_l" name="wheel_front_l" />

    <!-- Back Right Leg Actuators -->
    <general joint="leg_back_r_1" name="leg_back_r_1" />
    <general joint="leg_back_r_2" name="leg_back_r_2" />
    <velocity joint="wheel_back_r" name="wheel_back_r" />

    <!-- Back Left Leg Actuators -->
    <general joint="leg_back_l_1" name="leg_back_l_1" />
    <general joint="leg_back_l_2" name="leg_back_l_2" />
    <velocity joint="wheel_back_l" name="wheel_back_l" />
</actuator>
```

Software

Wheel Policy debug

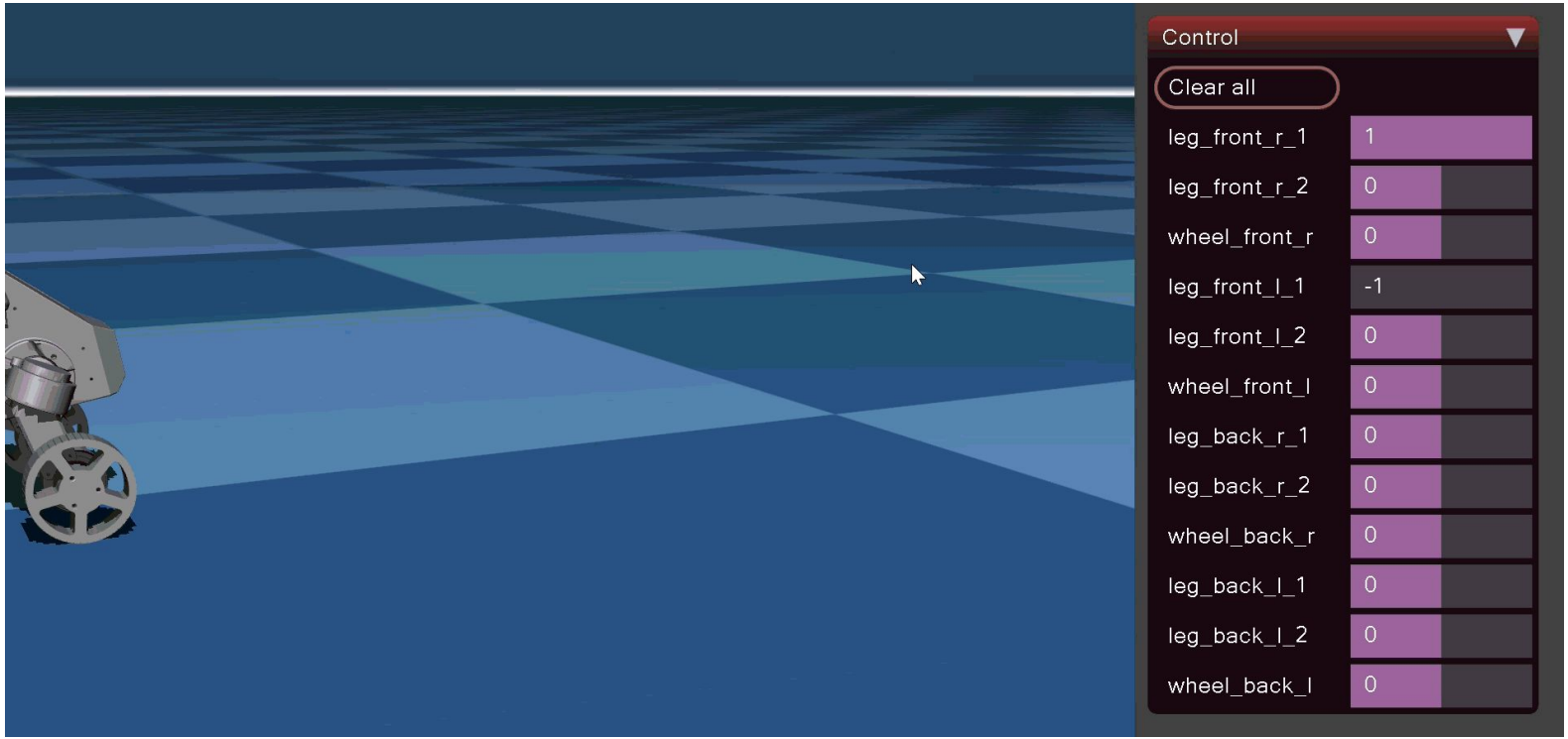
Changing from joint angle control to velocity control



Software

Wheel Policy debug

Changing from joint angle control to velocity control



Next Steps ⬆️

Next Steps ➡️ Hardware

- Mat & Patt are working on the heating element

Next Steps ➡️ Software

- Finalize wheel xml, Load Wheel with terrain and obstacle ⛰️

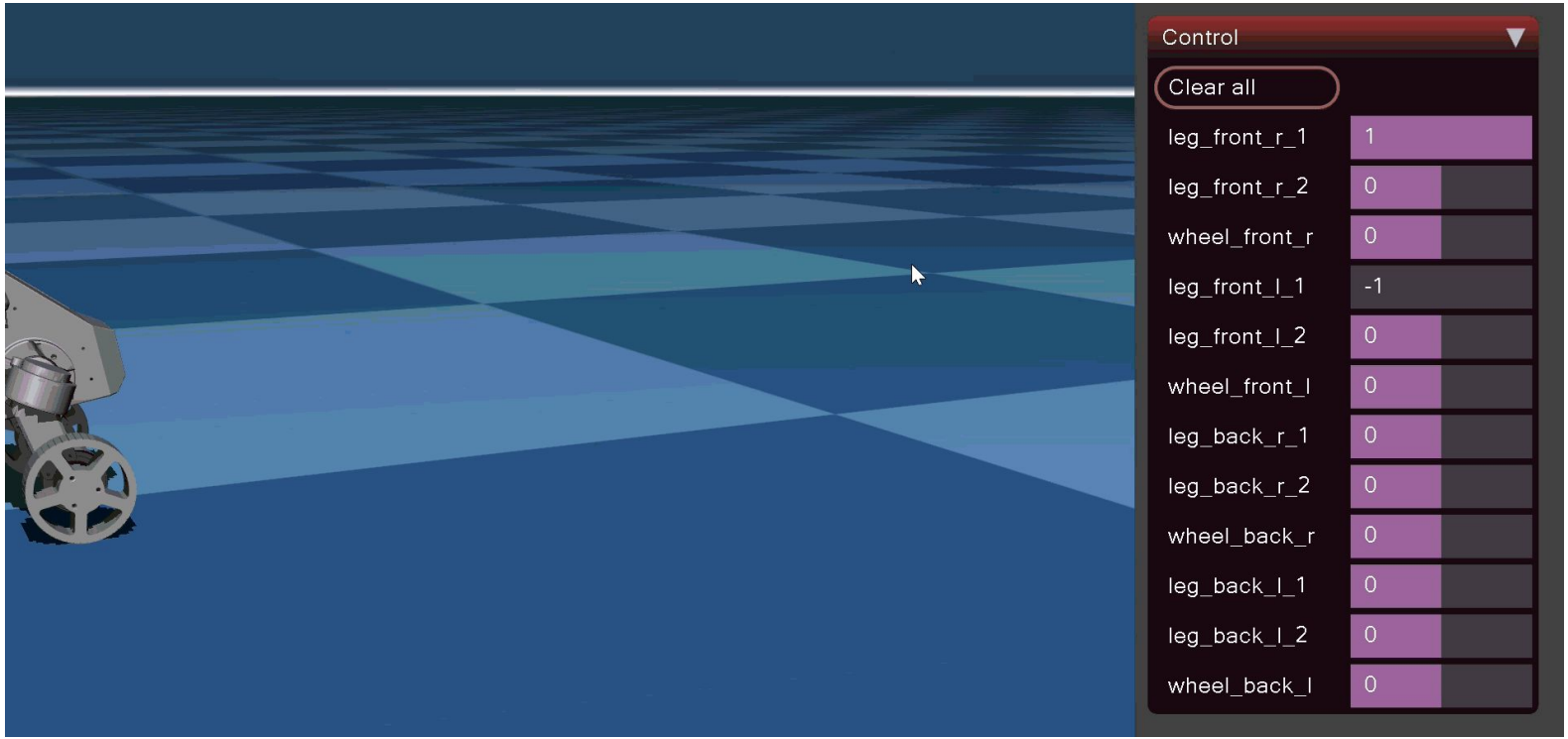
Pupper project

13/11/25 update
Tund

Software

Last week 🕰️

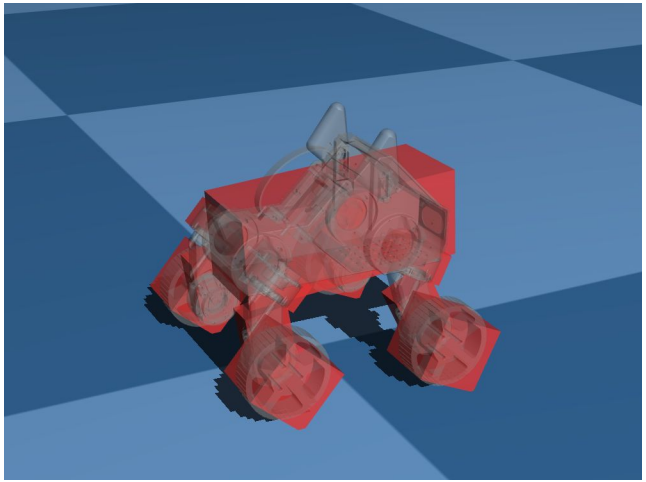
Inertia not setup correctly

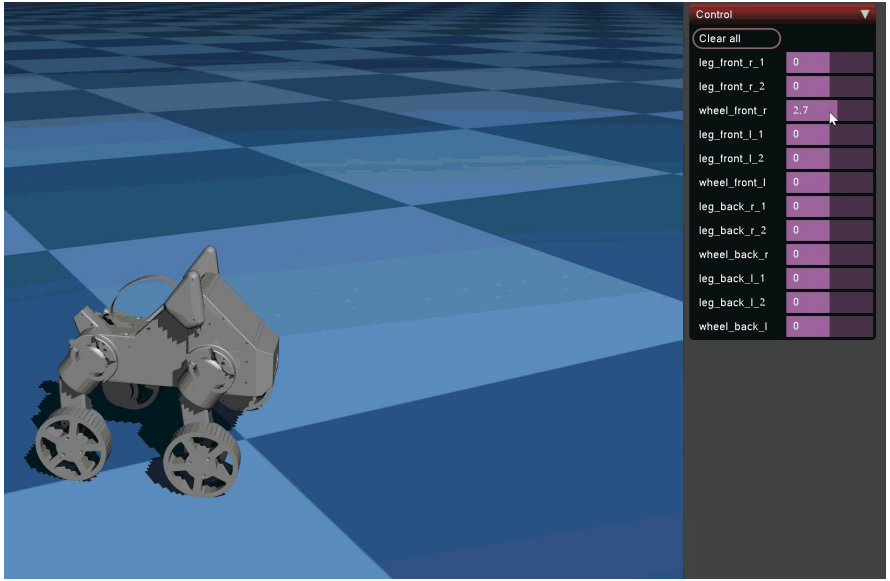


Software

Fixed ✅

Inertia not setup correctly

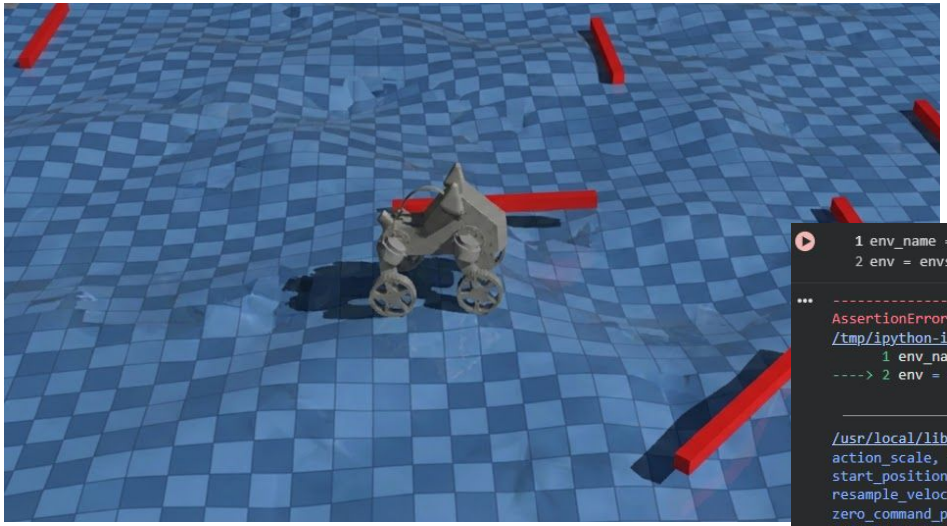




Software

Colab debug

Still having some problems running this new configuration on Colab ⚠️



```
1 env_name = 'pupper'
2 env = envs.get_environment(env_name, **env_kwargs)
```

```
---------------------------------------------------------------------------
AssertionError                            Traceback (most recent call last)
/tmp/ipython-input-2332918990.py in <cell line: 0>()
      1 env_name = 'pupper'
----> 2 env = envs.get_environment(env_name, **env_kwargs)

                                  1 frames
/usr/local/lib/python3.12/dist-packages/pupperv3_mjx/environment.py in __init__(self, path, reward_confi
action_scale, observation_history, joint_lower_limits, joint_upper_limits, dof_damping, position_control
start_position_config, foot_site_names, torso_name, upper_leg_body_names, lower_leg_body_names,
resample_velocity_step, linear_velocity_x_range, linear_velocity_y_range, angular_velocity_range,
zero_command_probability, stand_still_command_threshold, maximum_pitch_command, maximum_roll_command,
default_pose, desired_abduction_angles, angular_velocity_noise, gravity_noise, motor_angle_noise,
last_action_noise, kick_vel, kick_probability, terminal_body_z, early_termination_step_threshold,
terminal_body_angle, foot_radius, environment_timestep, physics_timestep, latency_distribution,
imu_latency_distribution, desired_world_z_in_body_frame, use_imu)
    208             mujoco.mj_name2id(sys.mj_model, mujoco.mjtObj.mjOBJ_SITE.value, f) for f in feet_sit
    209         ]
--> 210         assert not any(id_ == -1 for id_ in feet_site_id), "Site not found."
    211         self._feet_site_id = np.array(feet_site_id)
    212

AssertionError: Site not found.
```

Software

Going back to PPO environment 🔍

Guesses on what is causing problems

- **Actuator Conflict:** I suspect the wheels might be getting treated like leg position actuators, so position feedback could be fighting the velocity commands and causing the jittering.
- **Target Calculation:** It seems possible that the system is adding some default joint angle offsets to the velocity targets, which might be introducing an unintended bias.
- **Action Scale:** I think the action scaling for wheel velocities might be too small, limiting the achievable speed and making the robot look like it can't roll properly.

```
# override menagerie params for smoother policy
leg_indices = jp.array([0, 1, 3, 4, 6, 7, 9, 10])
wheel_indices = jp.array([2, 5, 8, 11])

# Set leg gains (PD control)
gainprm = sys.actuator_gainprm.at[leg_indices, 0].set(position_control_kp)
biasprm = sys.actuator_biasprm.at[leg_indices, 1].set(-position_control_kp)
biasprm = biasprm.at[leg_indices, 2].set(-dof_damping)

# Set wheel gains (Velocity control)
# Use position_control_kp as kv for now, but ensure no bias
gainprm = gainprm.at[wheel_indices, 0].set(position_control_kp)
biasprm = biasprm.at[wheel_indices, :].set(0.0)

sys = sys.replace(

    actuator_gainprm=gainprm,
    actuator_biasprm=biasprm,
```

Changing environment.py file 🌴

```
# Create vector action scale to allow higher velocity for wheels
self._action_scale = jp.full(12, action_scale)
wheel_indices = jp.array([2, 5, 8, 11])
# Set wheel action scale to 15.0 (rad/s) to utilize full control range
self._action_scale = self._action_scale.at[wheel_indices].set(15.0)
```

```
# Physics step
# For wheels (indices 2, 5, 8, 11), we want velocity control, so we don't add default_
# For legs, we want position control, so we add default_pose.
wheel_indices = jp.array([2, 5, 8, 11])
default_pose_for_calc = self._default_pose.at[wheel_indices].set(0.0)

motor_targets = default_pose_for_calc + lagged_action * self._action_scale
```

Pupper project

04/12/25 update
Tund

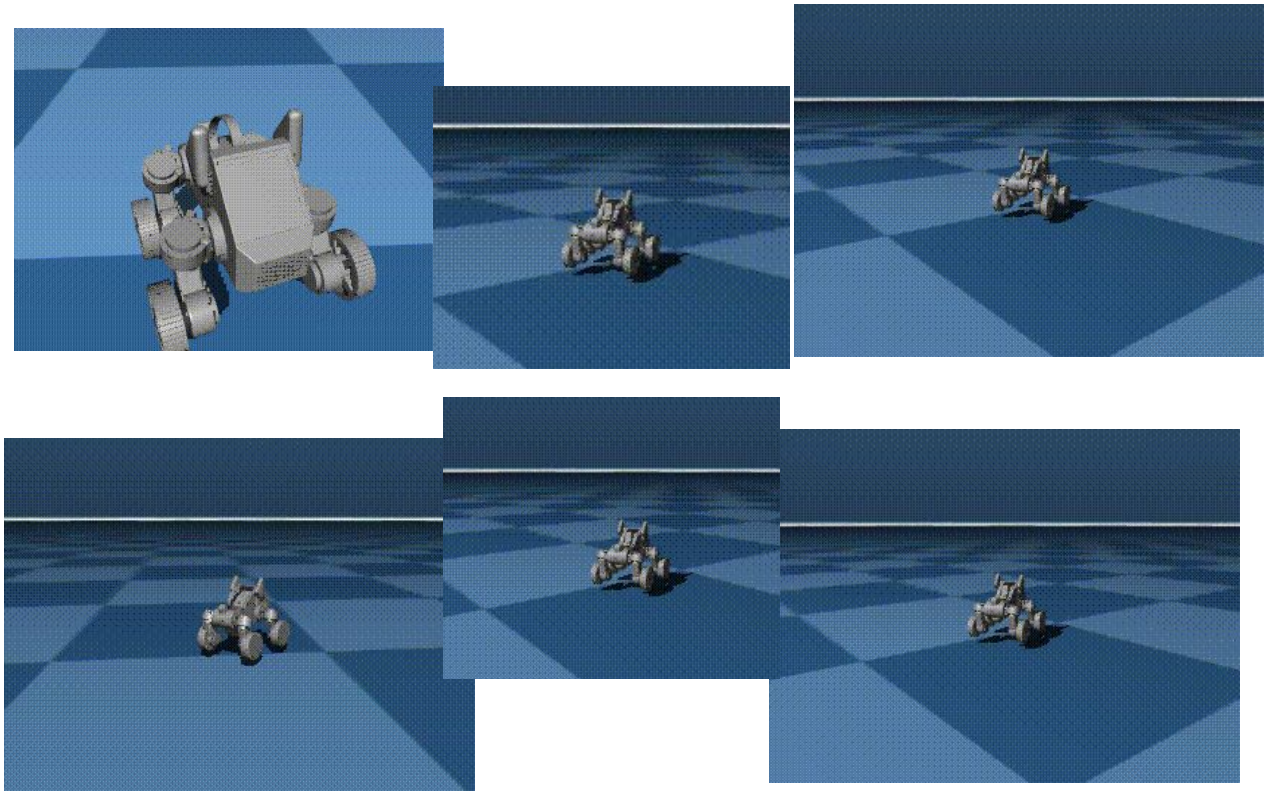
Software

Natural gait 

Stand still 

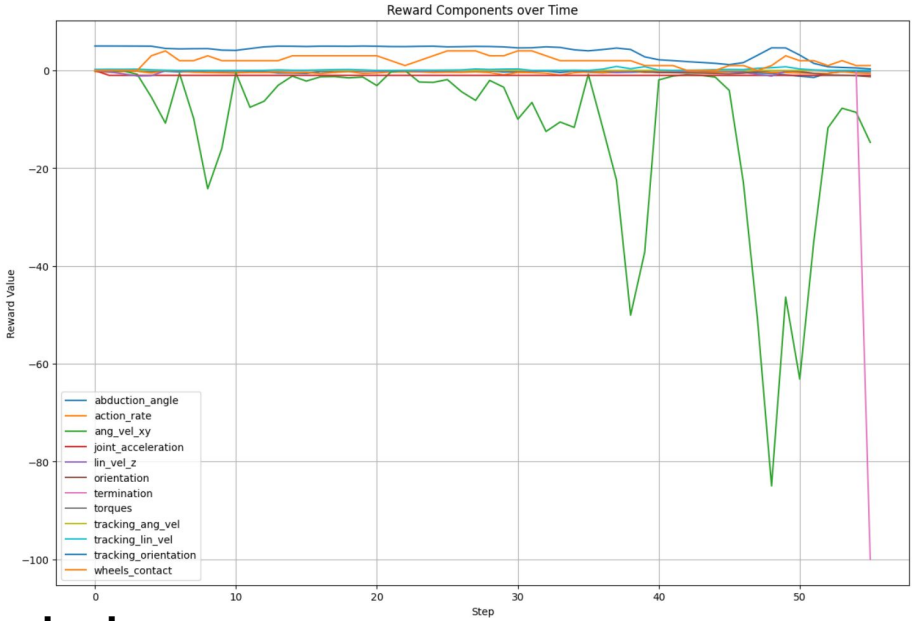
Software

Reward shaping needed 🔍



Software

```
--- Reward Summary ---
Reward Name                | Mean       | Min        | Max
-------------------------------------------------------------------------
abduction_angle            | -0.3210    | -1.4288    | -0.0000
action_rate                | -0.4019    | -0.9794    | -0.0757
ang_vel_xy                 | -11.5951   | -84.9833   | -0.0000
joint_acceleration         | -0.9821    | -1.0000    | -0.0000
lin_vel_z                  | -0.1671    | -1.1519    | -0.0004
orientation                | -0.1888    | -1.2805    | -0.0000
termination                | -1.7857    | -100.0000  | 0.0000
torques                    | -0.0090    | -0.0135    | -0.0000
tracking_ang_vel           | 0.0000     | 0.0000     | 0.0000
tracking_lin_vel           | 0.1568     | 0.0004     | 0.8169
tracking_orientation       | 3.8919     | 0.3196     | 5.0000
wheels_contact             | 2.0000     | 0.0000     | 4.0000
-------------------------------------------------------------------------

--- State Summary ---
Metric                     | Mean       | Max
-------------------------------------------------------------------------
linear_vel                 | 0.4009     | 0.7775
angular_vel                | 5.9052     | 11.8113
z_height                   | 0.1306     | 0.1694
-------------------------------------------------------------------------
```



The "Root Cause": Why the Episode Crashed

- **Visual:** Look at the **Top Right Graph (Reward Components)**.
- **Observation:** The Green line (`ang_vel_xy`) crashes down to **-80** at step 55, and the lines cut off.
- **What it means:** The simulation terminated early (at step 55 out of 1000). The violent shaking triggered such a massive penalty for "unstable rotation" (`ang_vel_xy`) that the episode likely hit a termination threshold.

Software

Deep Dive: Infrastructure & Dependencies

The Problem: 'Dependency Hell' 🪰

- Colab runtime update broke JAX compatibility.
- Downgrading JAX created cascading conflicts with `orbax-checkpoint` and `optax`.
- Result: Unstable environment, `pip` solver failures.

The Investigation: Surgical Dependency Pinning

- Identified last known stable configuration in `wheelcolab.ipynb`.
- Located compatible package versions for a core JAX release.

```
# wheelcolab.ipynb
!pip install jax==0.4.30
!pip install orbax-checkpoint==0.5.9
!pip install optax==0.2.2
```

Package	Old Value	New Value	Reason
`jax`	`latest`	`0.4.30`	Core stable release (June 2024)
`orbax-checkpoint`	`latest`	`0.5.9`	Required by trainer, compatible with JAX 0.4.x
`optax`	`latest`	`0.2.2`	Required by PPO, compatible with JAX 0.4.x

Status: Fixed ✅

Software

Deep Dive: Infrastructure & Dependencies

The Problem: Extreme Simulation Lag ⚠️

- Debugging cycles took ~10 minutes.
- Inefficient "eager execution" mode: Python sent each physics step as a separate instruction to the GPU.
- Wasted Colab compute units.

The Investigation: JIT Compilation


```python
# environment.py

@jax.jit
def step(state, action):
    # ... thousands of physics operations ...
    return next_state
```

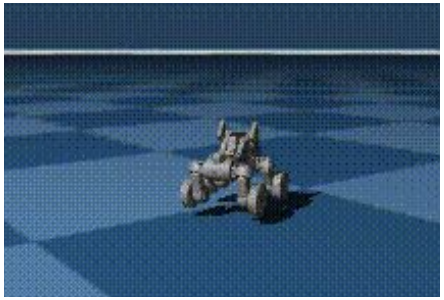
Result: Debug cycle time reduced from ~10 min to < 10 sec.

Pupper project

18/12/25 update
Tund

Software

Hardware representation 🔍



```
<!-- Front Right Leg Actuators -->
<general joint="leg_front_r_1" name="leg_front_r_1" />
<general joint="leg_front_r_2" name="leg_front_r_2" />
<velocity joint="wheel_front_r" name="wheel_front_r" kv="1" ctrlrange="-40 40"/>

<!-- Front Left Leg Actuators -->
<general joint="leg_front_l_1" name="leg_front_l_1" />
<general joint="leg_front_l_2" name="leg_front_l_2" />
<velocity joint="wheel_front_l" name="wheel_front_l" kv="1" ctrlrange="-40 40" />

<!-- Back Right Leg Actuators -->
<general joint="leg_back_r_1" name="leg_back_r_1" />
<general joint="leg_back_r_2" name="leg_back_r_2" />
<velocity joint="wheel_back_r" name="wheel_back_r" kv="1" ctrlrange="-40 40" />

<!-- Back Left Leg Actuators -->
<general joint="leg_back_l_1" name="leg_back_l_1" />
<general joint="leg_back_l_2" name="leg_back_l_2" />
<velocity joint="wheel_back_l" name="wheel_back_l" kv="1" ctrlrange="-40 40" />
```

Potential problem spots

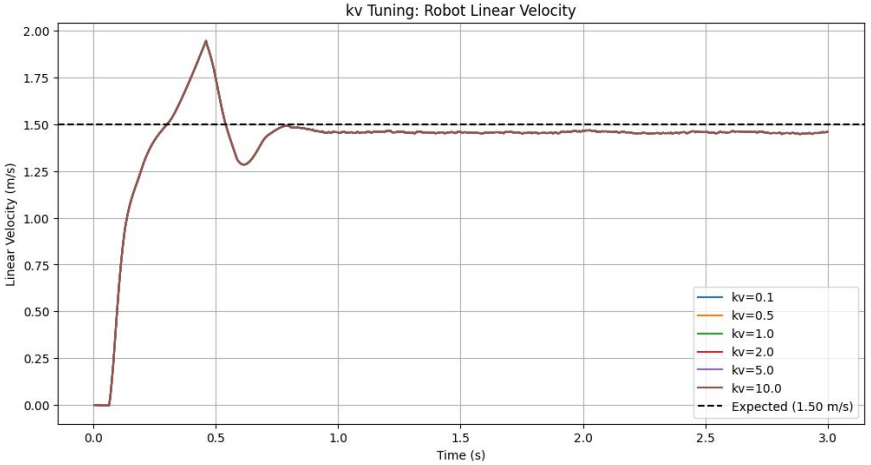
- **Contact model:** Was using spheres due to it being easier to program -> now using more accurate cylinders
- **KV gains:** Jittery motion could be cause by gains being too low currently set to 1 by default
- **Control range:** It seems possible that the system is adding some default joint angle offsets to the velocity targets, which might be introducing an unintended bias.

Software

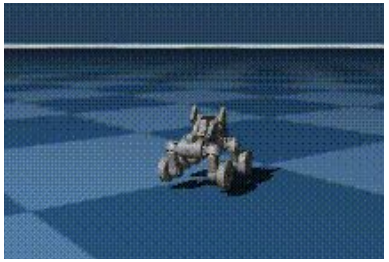
Hardware calibration 🔍

Potential problem spots: KV gains

Goal: Step input 15 m/s
Testing: (0.1, 1.0, 5.0, 10.0)



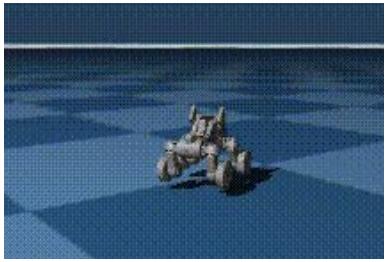
kv	Steady State (m/s)	Rise Time (s)	Error (m/s)
0.1	1.4550	0.2120	0.0450
0.5	1.4550	0.2120	0.0450
1.0	1.4550	0.2120	0.0450
2.0	1.4550	0.2120	0.0450
5.0	1.4550	0.2120	0.0450
10.0	1.4550	0.2120	0.0450



```
<!-- Front Right Leg Actuators -->
<general joint="leg_front_r_1" name="leg_front_r_1" />
<general joint="leg_front_r_2" name="leg_front_r_2" />
<velocity joint="wheel_front_r" name="wheel_front_r" kv="1" ctrlrange="-40 40"/>

<!-- Front Left Leg Actuators -->
<general joint="leg_front_l_1" name="leg_front_l_1" />
<general joint="leg_front_l_2" name="leg_front_l_2" />
<velocity joint="wheel_front_l" name="wheel_front_l" kv="1" ctrlrange="-40 40" />

<!-- Back Right Leg Actuators -->
<general joint="leg_back_r_1" name="leg_back_r_1" />
<general joint="leg_back_r_2" name="leg_back_r_2" />
<velocity joint="wheel_back_r" name="wheel_back_r" kv="1" ctrlrange="-40 40" />

<!-- Back Left Leg Actuators -->
<general joint="leg_back_l_1" name="leg_back_l_1" />
<general joint="leg_back_l_2" name="leg_back_l_2" />
<velocity joint="wheel_back_l" name="wheel_back_l" kv="1" ctrlrange="-40 40" />
```

Software

Hardware calibration 🔍



Potential problem spots: Control range

Goal: Max speed requirement 1 m/s
Testing: Does current ctrlrange of ±15 rad/s. Meet those requirements

$$r = 0.04445 \text{ m}$$

$$v = 15 \text{ rad/s} \cdot 0.04445 \text{ m} \approx 0.66675 \text{ m/s}$$

$$\omega = \frac{v}{r}$$

$$\omega = \frac{1.5 \text{ m/s}}{0.04445 \text{ m}} \approx 33.745 \text{ rad/s}$$

$$\texttt{ctrlrange} = [-40, 40] \text{ rad/s}$$

```
<!-- Front Right Leg Actuators -->
<general joint="leg_front_r_1" name="leg_front_r_1" />
<general joint="leg_front_r_2" name="leg_front_r_2" />
<velocity joint="wheel_front_r" name="wheel_front_r" kv="1" ctrlrange="-40 40"/>

<!-- Front Left Leg Actuators -->
<general joint="leg_front_l_1" name="leg_front_l_1" />
<general joint="leg_front_l_2" name="leg_front_l_2" />
<velocity joint="wheel_front_l" name="wheel_front_l" kv="1" ctrlrange="-40 40" />

<!-- Back Right Leg Actuators -->
<general joint="leg_back_r_1" name="leg_back_r_1" />
<general joint="leg_back_r_2" name="leg_back_r_2" />
<velocity joint="wheel_back_r" name="wheel_back_r" kv="1" ctrlrange="-40 40" />

<!-- Back Left Leg Actuators -->
<general joint="leg_back_l_1" name="leg_back_l_1" />
<general joint="leg_back_l_2" name="leg_back_l_2" />
<velocity joint="wheel_back_l" name="wheel_back_l" kv="1" ctrlrange="-40 40" />
```

Rotation Loophole

By spinning in place, the robot could:

- Satisfy orientation or stability rewards
- Avoid falling immediately
- Accumulate *more reward* than by attempting impossible forward motion

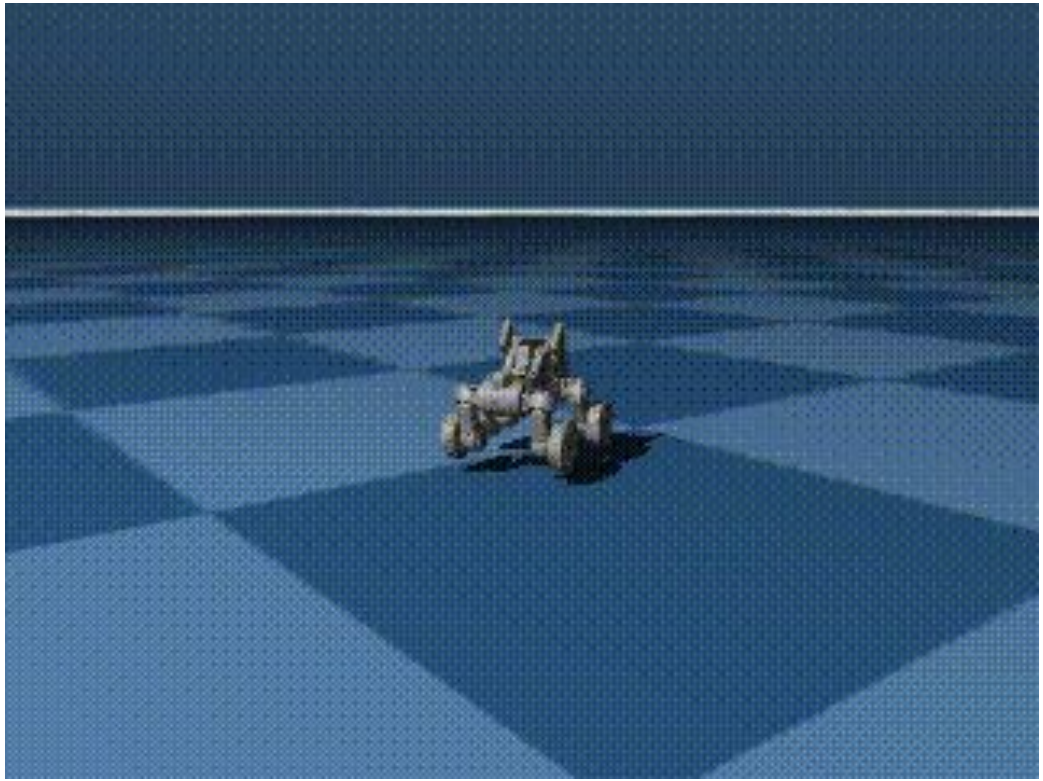
As a result, **spinning became the locally optimal strategy**.

Software

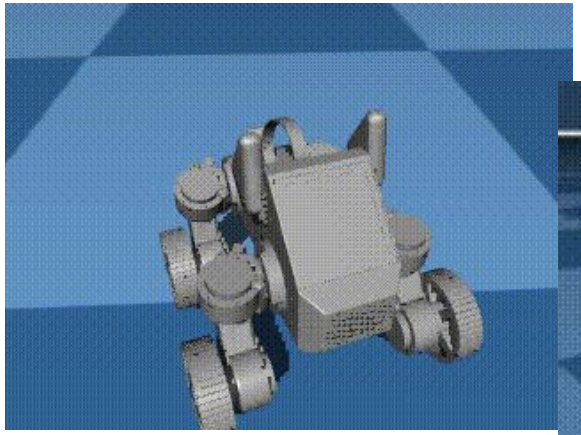
Hardware calibration 🔍

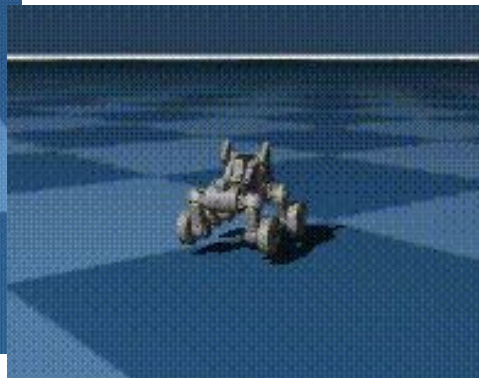
Results after changes:

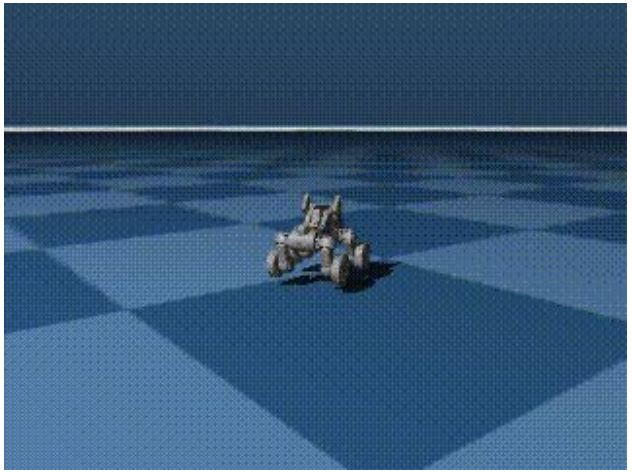
- Bouncy and Jittery indicating that the robot is still cheating in someway.
- Will look into action rate, joint acceleration, and body orientation reward shaping to try fix this

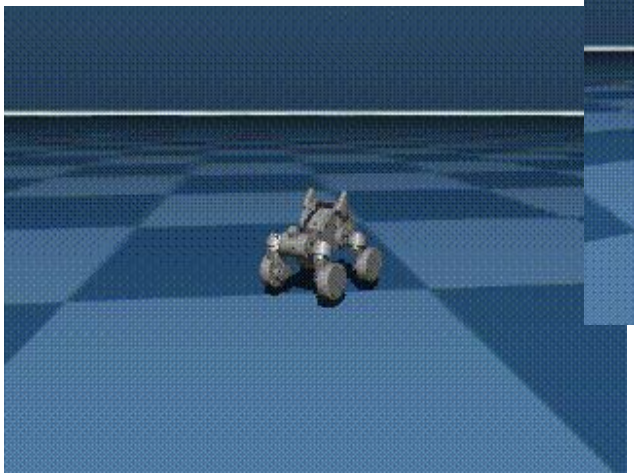


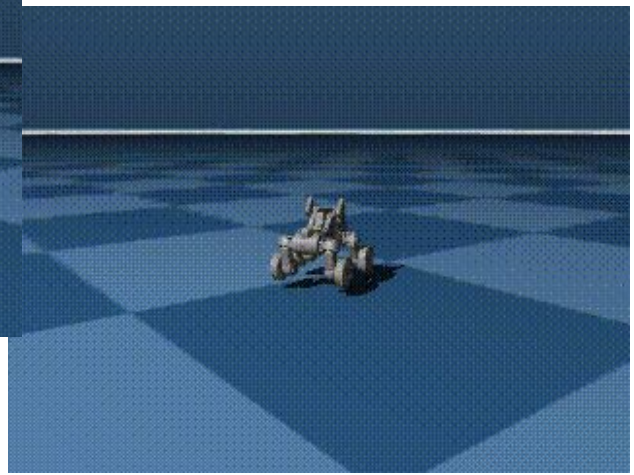
Software









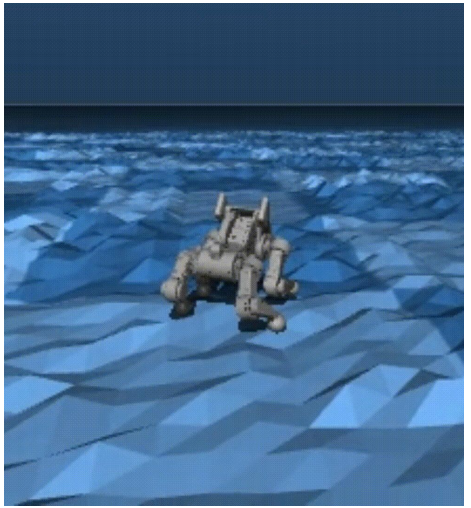


Pupper project

08/01/26 update
Tund

Systems

System level understanding 🔎



Policy/control timestep: `dt = 0.02s`

Command sampling: `250 steps × 0.02 s = 5.0s`

Episode length: `500 steps × 0.02 s = 10.0s`

Training environment number = `8192`

Action Transformation: The policy learns to slightly modify a nominal standing pose (PPO)

- `Targets_position = default_pose + (action * action_scale)`

Target generated by PPO

Policy/control timestep: `dt = 0.02 s`

How the legs are controlled:

- `Torque = Kp * (Target position - Current_Position) - Kv * Current_Velocity`

5

0.1

Environment timesteps = `300 million`

Systems

System level understanding 🔎

Action Transformation: The policy learns to slightly modify a nominal standing pose (PPO)

- `Targets_position = default_pose + (action * action_scale)`

Target generated by PPO

Policy/control timestep: `dt = 0.02 s`

How the legs are controlled:

- `Torque = Kp * (Target position - Current_Position) - Kv * Current_Velocity`

5

0.1

Position control is problematic for wheel

2 options in mujoco:

- **Velocity control**
- **motor/torque control**