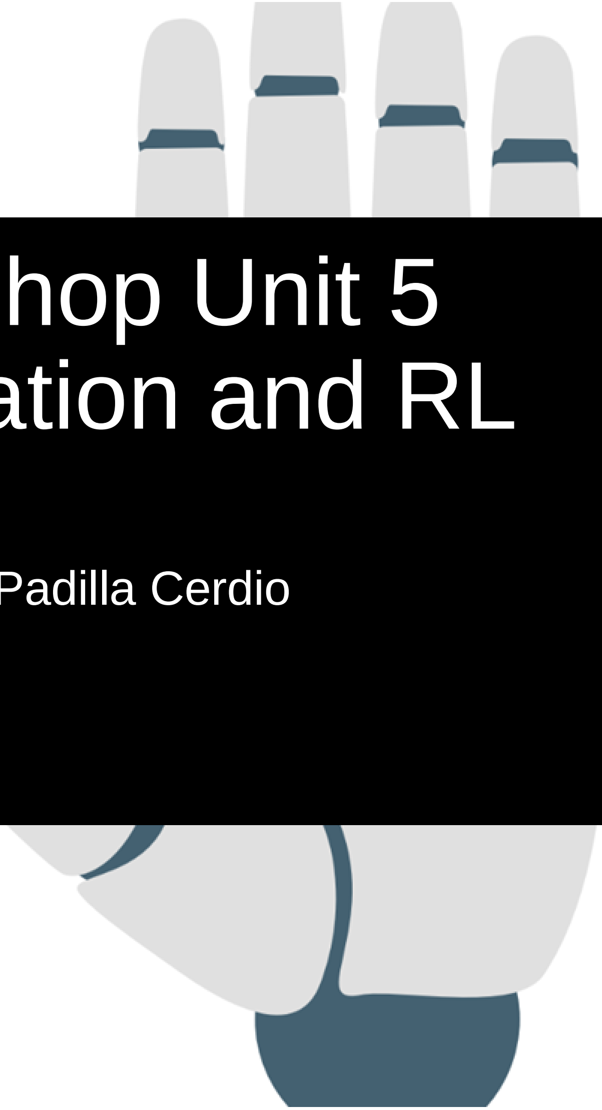
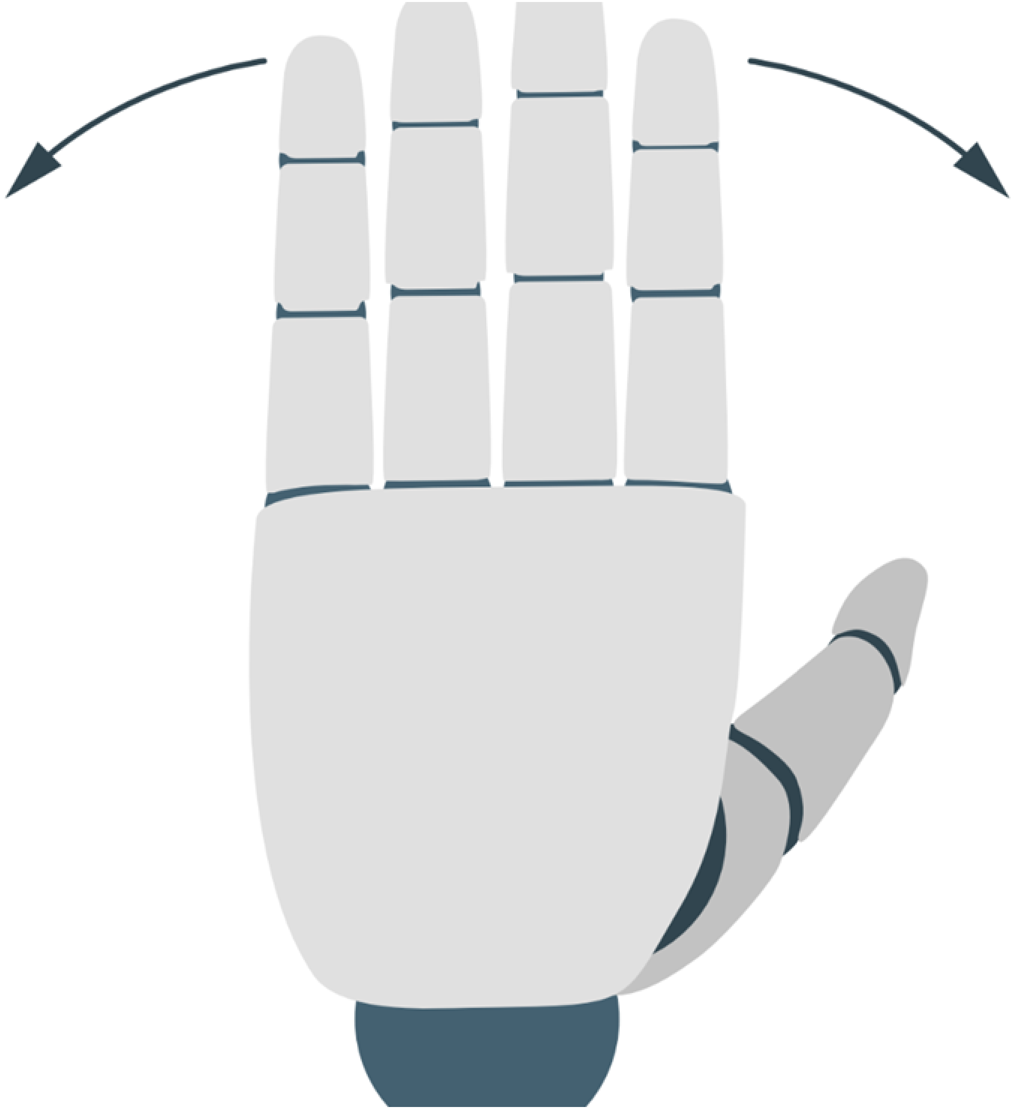
Workshop Unit 5
Simulation and RL

TA Esteban Padilla Cerdio

20.10.2025





Isaac Lab and IsaacSim

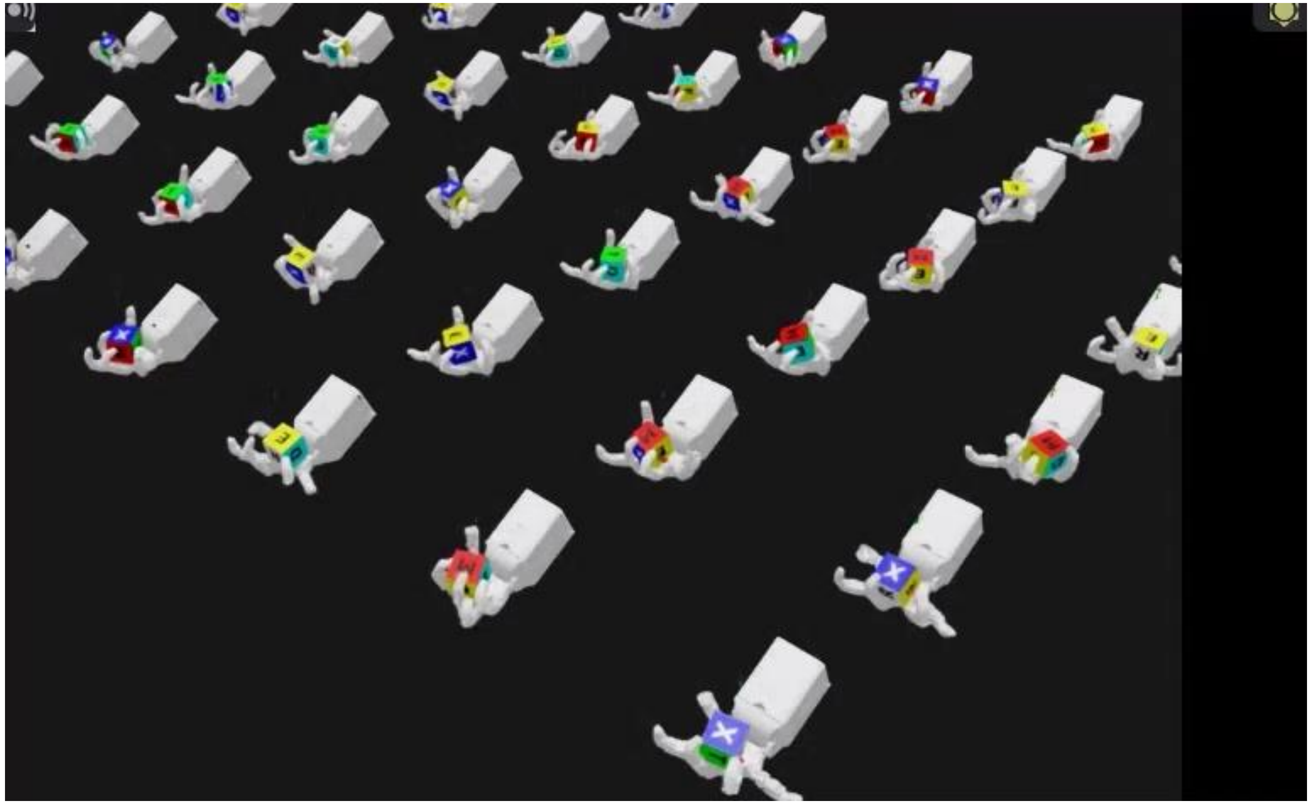
Isaac Lab is a robotics research platform built on IsaacSim

GPU Accelerated physics simulation with PhysX

Massively parallel environments

How does this help train RL policies better?

Works with URDF, MJCF and CAD



Robot description files

- **Unified Robot Description Format (URDF)**
- **MuJoCo XML Modeling File (MJCF)**

- XML Files that describe the structure and characteristics of the robot
- What type of joints does it have?
- Which links does each joint connect?
- How much can these joints move?
- How much does each link measure and weigh?

- Includes the CAD files so that it can be visualized in simulation environments

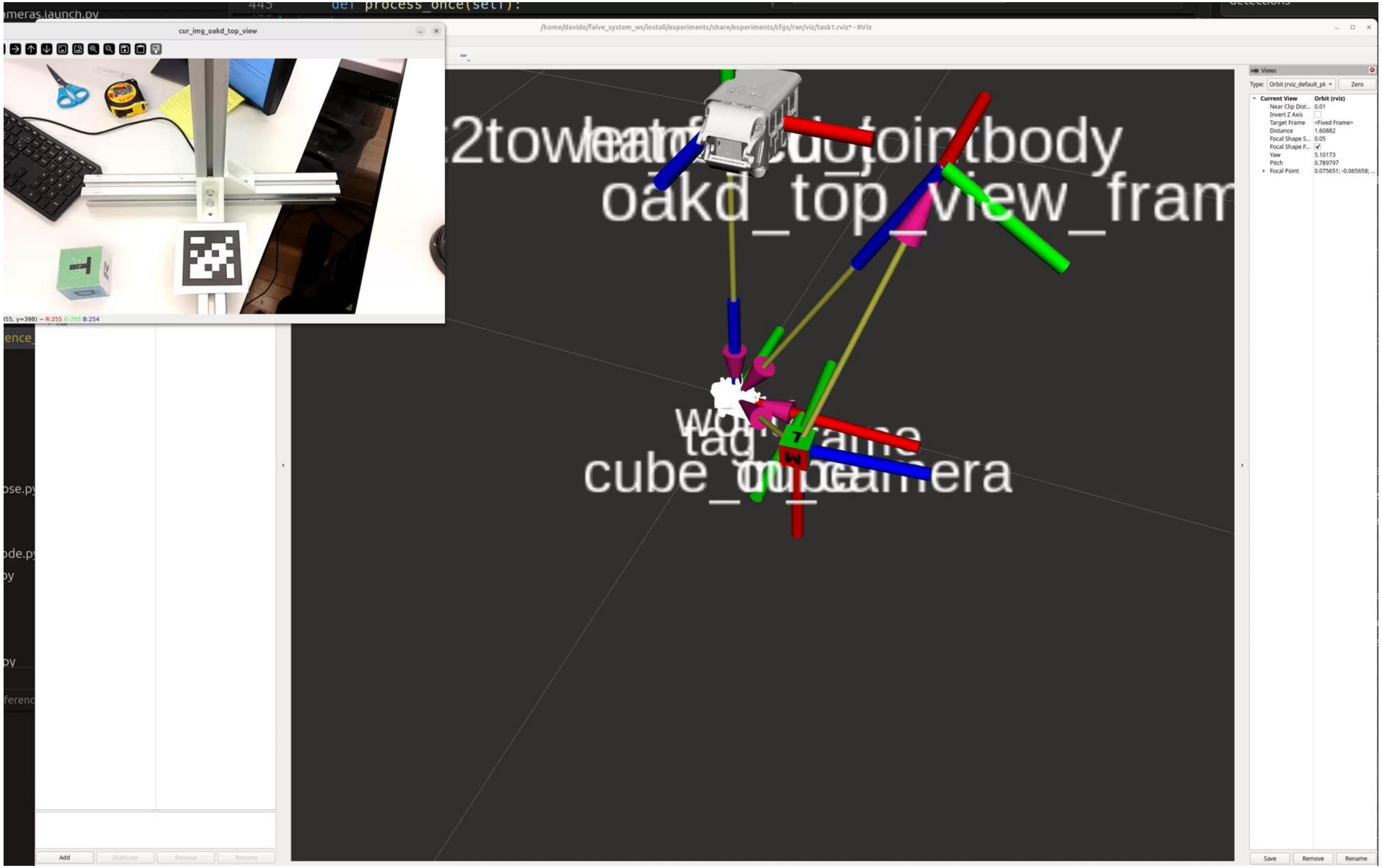
- **Universal Scene Description**

- Extends the description to 3D scenes instead of single models

- **If you modify your hand, you need to modify the corresponding files before training**

```
<joint type="revolute" name="right_thumb_abd">
  <parent link="right_thumb_mp"/>
  <child link="right_thumb_abd_jointbody"/>
  <origin xyz="6.938893903907228e-18 0.0 0.0" rpy="-1.22173 0.0 0.0"/>
  <axis xyz="0.0 0.34202085907197166 0.9396923602752502"/>
  <limit lower="-1.08211" upper="0.0" effort="100" velocity="100"/>
</joint>
<joint type="fixed" name="right_thumb_abd_offset">
  <parent link="right_thumb_abd_jointbody"/>
  <child link="right_thumb_pp"/>
  <origin xyz="-0.03 -0.0 -0.0" rpy="0.0 0.0 0.0"/>
</joint>
<link name="right_thumb_ip">
  <inertial>
    <origin xyz="0.00135 2e-05 0.01383" rpy="0.000370000000000000001 0.032630000
    <mass value="0.01402"/>
    <inertia ixx="1.7e-06" iyy="1.64e-06" izz="5.9e-07" ixy="0" ixz="0" iyz="0
  </inertial>
  <visual name="right_visual_thumb_ip_mesh">
    <origin xyz="0.0012998948492342203 7.159773213648307e-06 0.013619877706625
    <geometry>
      <mesh filename="package://orcahand_description/assets/urdf/right/visual
    </geometry>
    <material name="white"/>
  </visual>
  <collision name="right_collision_thumb_ip_mesh">
    <origin xyz="0.0013560375396256981 0.00012710825491742077 0.01344632766073
    <geometry>
      <mesh filename="package://orcahand_description/assets/urdf/right/collis
    </geometry>
```

RVIZ



Amazon Web Services (AWS)

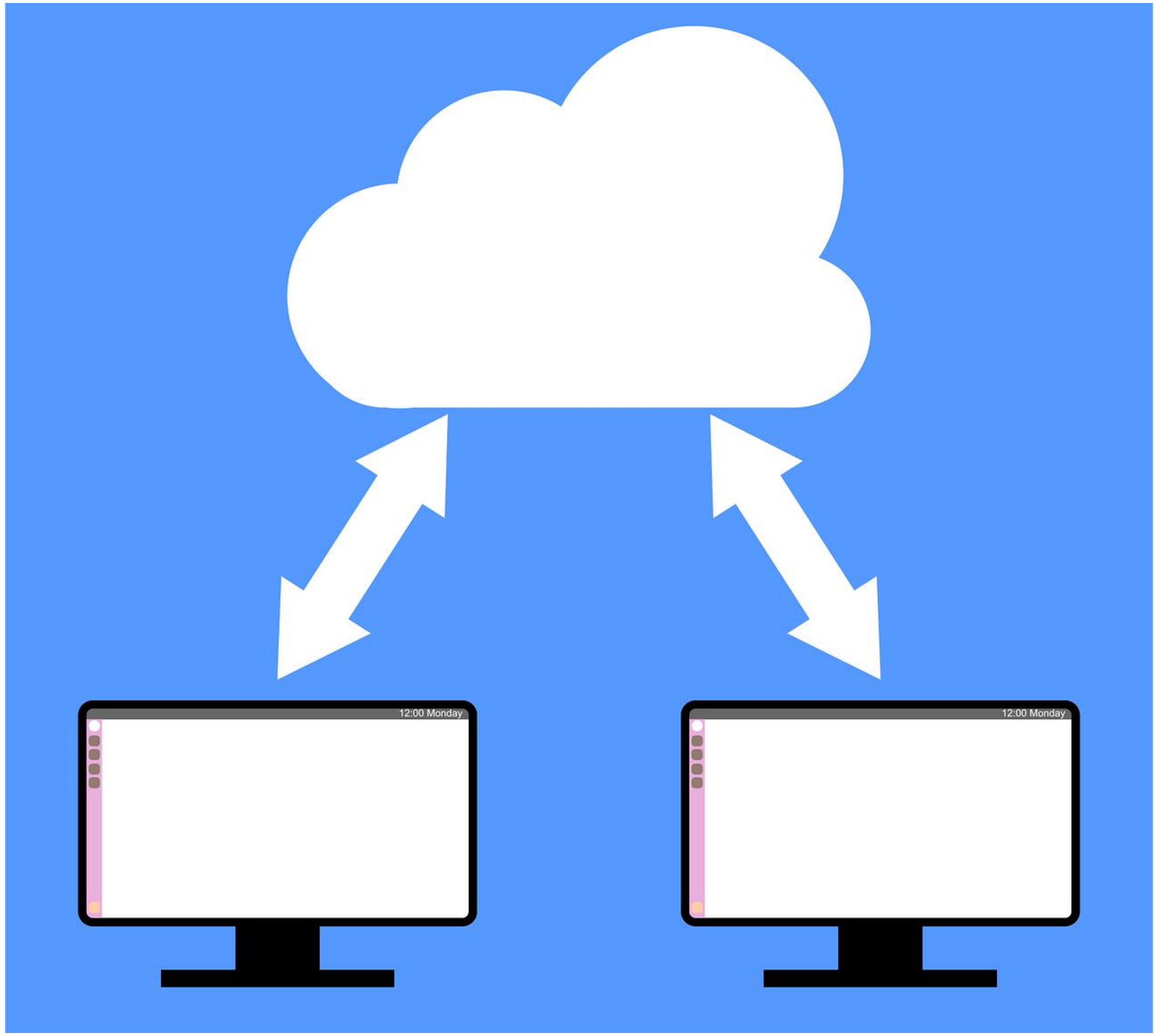
Cloud-computing platform

Personal computing instance with powerful GPUs to use for training and simulation

Accessible remotely via SSH

Turns off only after lack of usage. You can leave it "running" for hours at a time.

If you have the power locally, you're welcome to use it



Connecting through SSH

Access the SRL AWS Gateway

ssh -i ~/.ssh/srl_aws.pem ec2-user@3.126.230.101

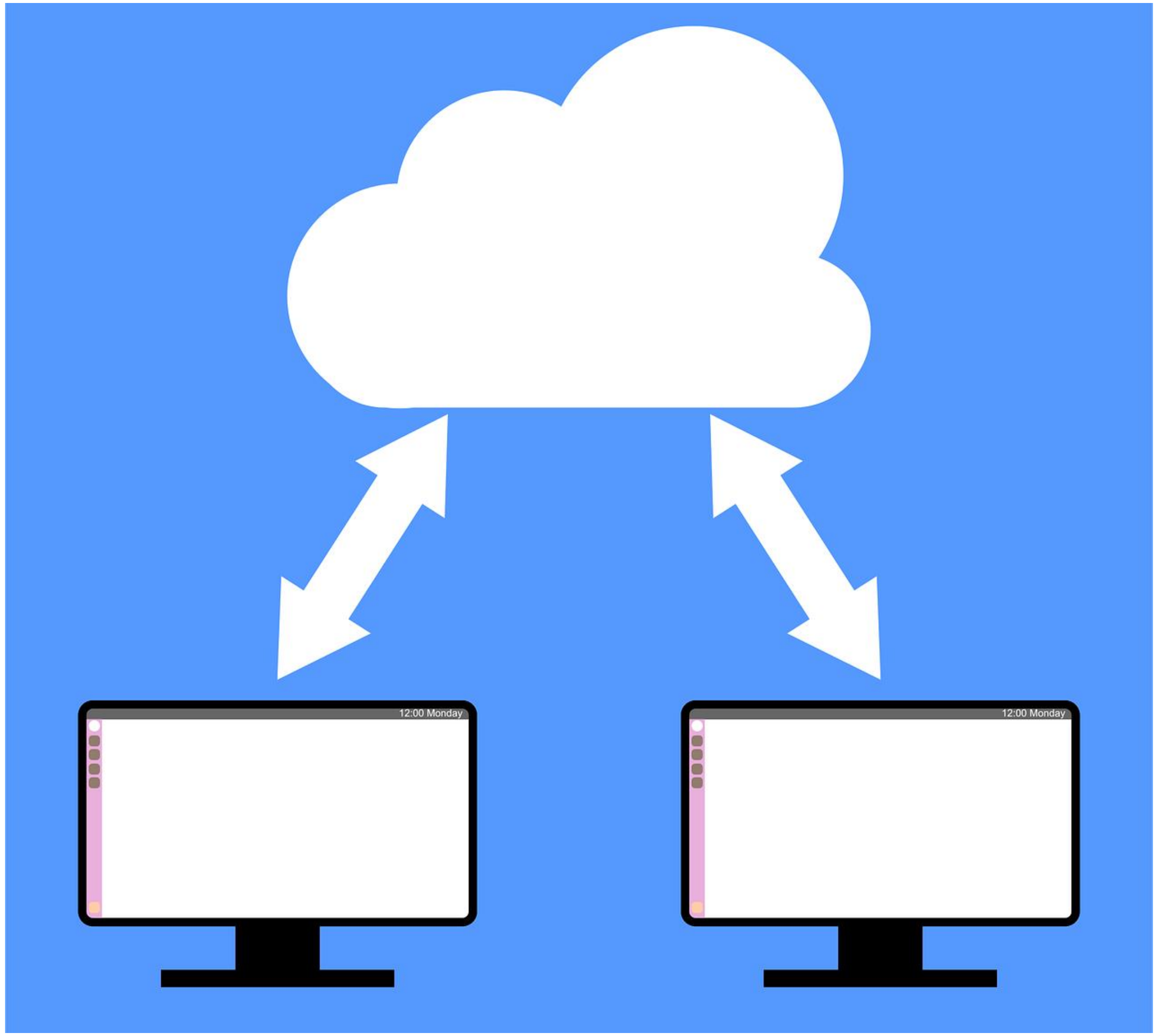
Kick-start your instance if it is not already running

python3 aws-gateway/client.py start <username>
stop
status
reboot

SSH into your instance

ssh -i ~/.ssh/srl_aws.pem ubuntu@[address of personal EC2 instance]

Optional: Access the filesystem with VSCode Remote-SSH



Sidetrack: Conda

Self-contained environments for your projects

Lets you choose a different Python version and parameters per project

Anything that you install will be installed only for that environment

If you mess up, you can always delete and restart without affecting your whole system

Make sure you use Conda to install IsaacLab and faive_lab in the same environment

Installing Isaac on your AWS environment

Follow the steps in the documentation

- Be sure to use the proper Conda environment with Python 3.11

Install the corresponding Torch packages

- Here is where Cuda hell usually starts

Install IsaacSim as a Python module with Pip

Install IsaacLab (maybe by forking SRL's own implementation)

FAIVE LAB

Our own environment for IsaacLab-based projects

Includes all the necessary setup for the ORCA Hand

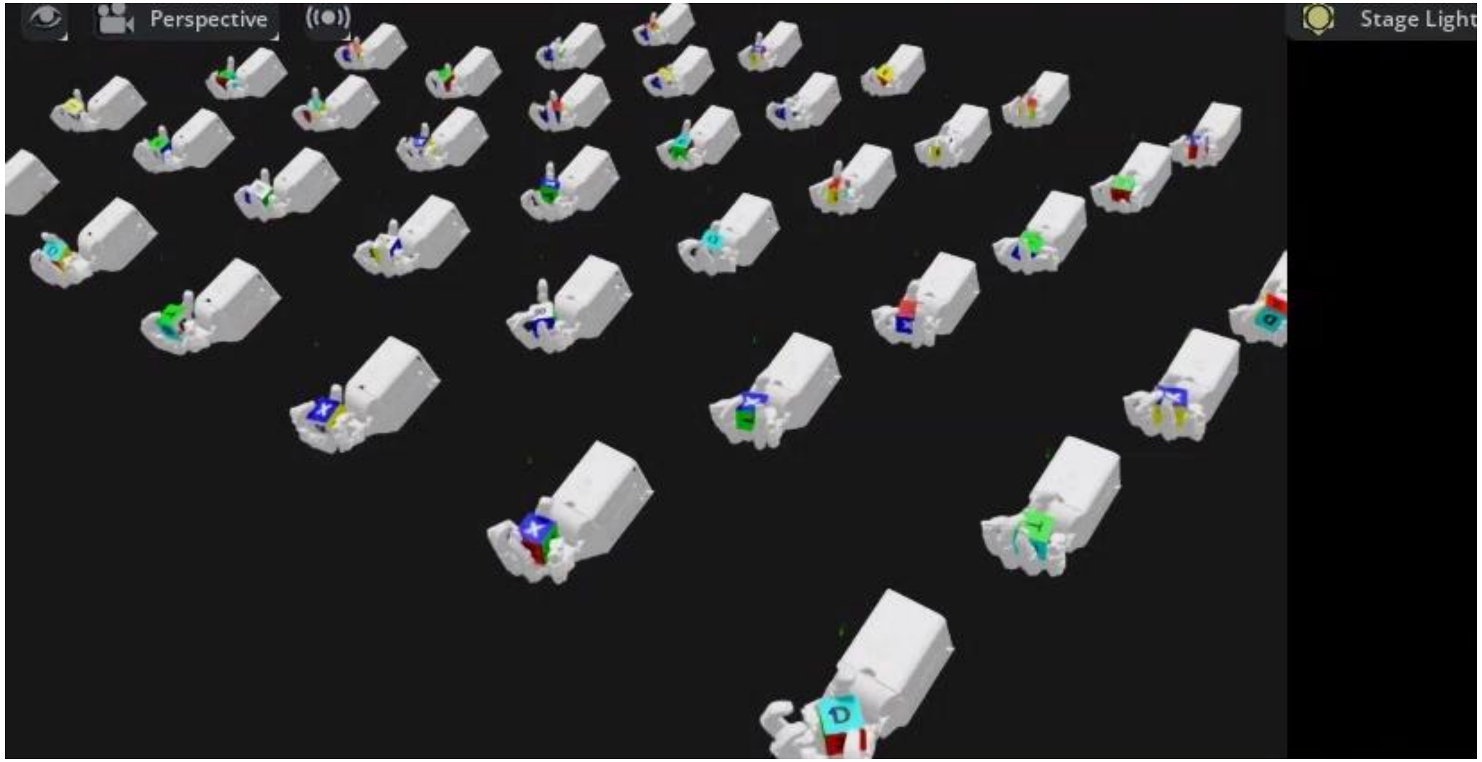
Isolated environment for RL Training

Fork it from GitHub on your AWS instance and follow the installation guide

SoftRobotics Laboratory

Training a policy

```
python3 scripts/rsl_rl/train.py  --task=orca-inhand-repose-v0 --logger wandb  --headless
--num_envs=8192 --video --video_interval=4000
```



Viewing the simulation

Option 1:

Run your training without

--headless

Let IsaacSim launch and show the “gym” environment

Connect with Amazon DCV remote desktop to view

The simulation doesn’t need to be visible for it to be running

Be patient, it takes a while to load.

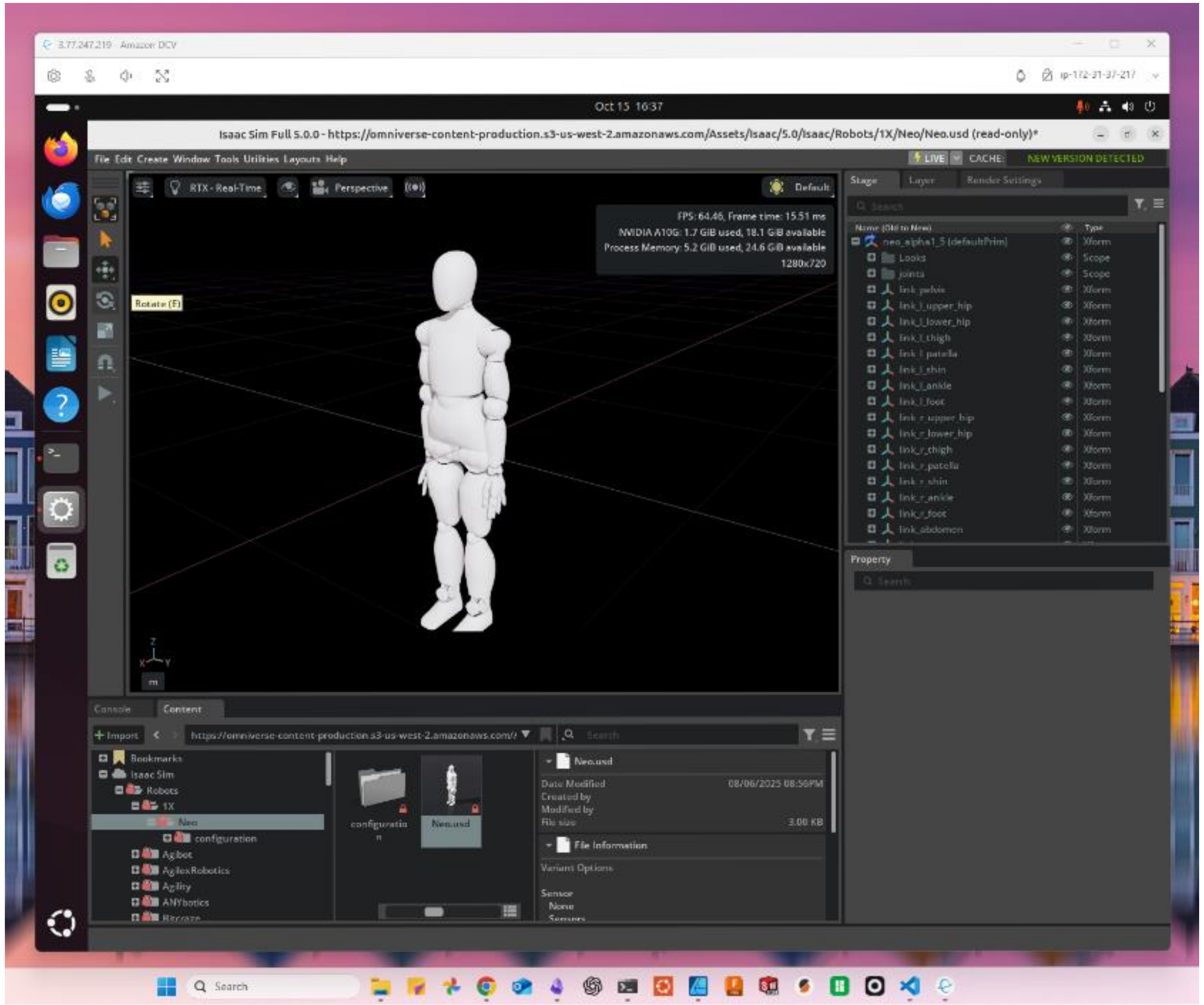
Option 2:

Record videos every N episodes

--video=True

--video_interval=2000

Transfer the videos through SCP or view them through VSCode SSH



Logging and Monitoring with Weights and Biases

AI Developer platform for tracking training

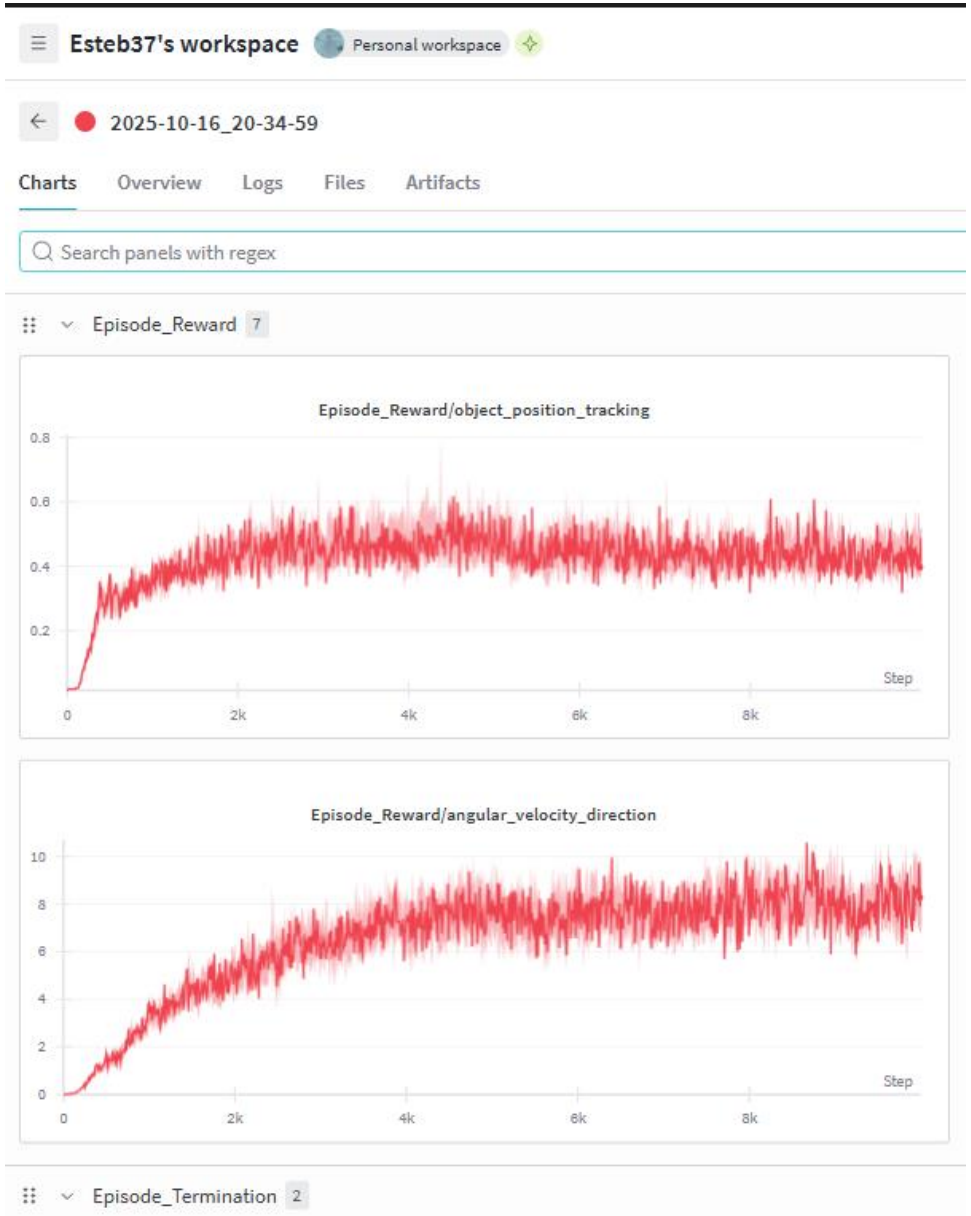
View terminal log history with timestamps

View progress in rewards, losses, mean episode length, etc

Share data with your team

Compare different runs

wandb.ai



Sim2Real Gap

Rarely will something you train in simulation be perfectly suited for real life

Still one of the biggest problems in robotics

It’s your job to bridge this gap as much as possible

(a) (b)

Domain Randomization

Vary elements in the simulation randomly during training

- Visual cues (lighning, textures)
- Physics (friction, damping)
- Noise

Prevent the policy from learning patterns related to the simulation

Forces the policy to focus on invariant features

Adds robustness and generalization

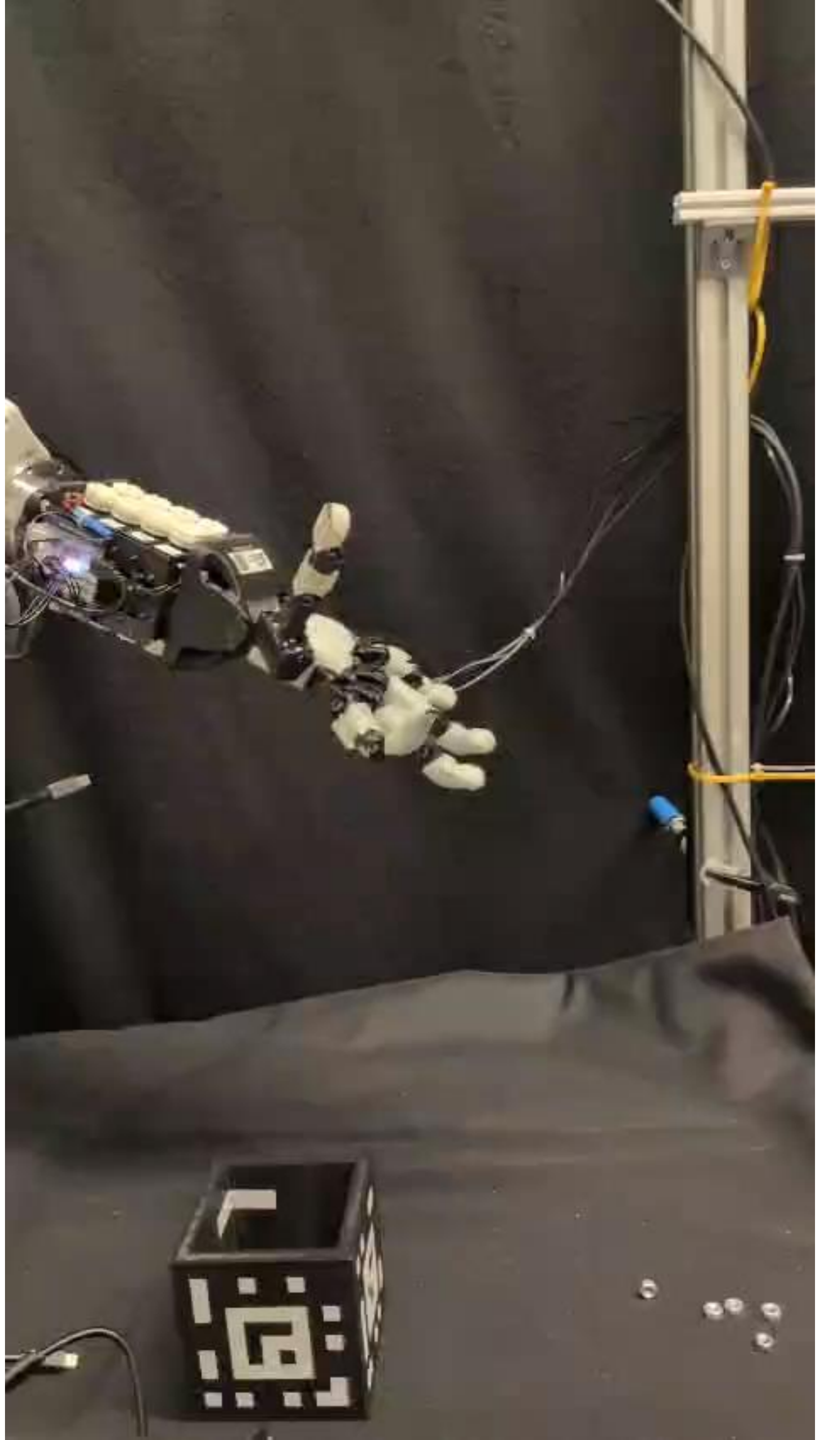
Avoids **overfitting**

System Identification

Running experiments on both your real hardware and your simulated software to find values on physical characteristics

- Friction
- Damping
- Stiffness

The resulting simulation captures the real dynamics more accurately

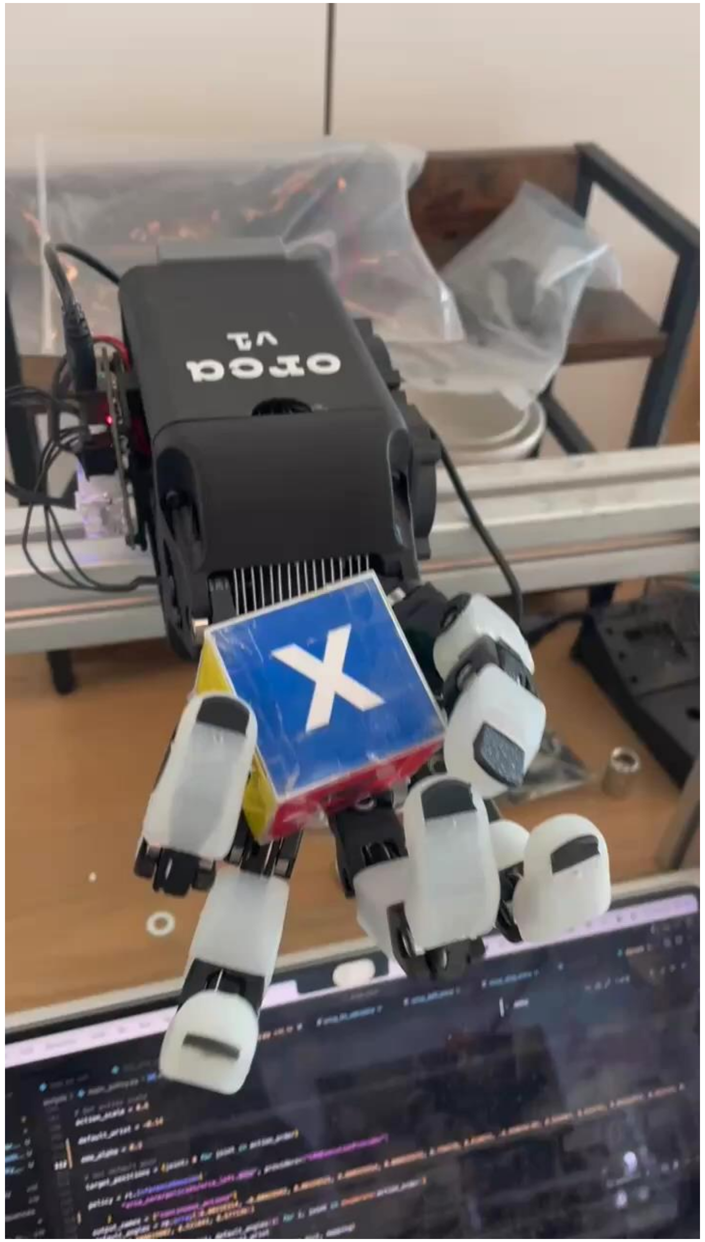


Real World Deployment

Your training session will produce a PyTorch model

Inference Node

We will give you a guide on how to deploy this model on the workstation after you've connected your own hands



Task 1 breakthrough and evaluation

1. Setup training of a proprioception-only in-hand cube reorientation around the z-axis
2. Deploy the trained policy to the physical Orca hand
3. Implement domain randomization and system identification to improve sim2real transfer
4. Train a policy that uses the cube pose from the camera and deploy it to the real world
5. (bonus) Train and deploy a policy that can reorient the cube to any given pose

Deadline: November 10th

Each team will showcase both the proprioception-only policy and the cube pose feedback policy during the task 1 showcase.

Each deployed policy will be evaluated with a score that is a function of average rotational speed and amount of rotation before falling. Score will be averaged between 10 runs

Minimum passing criteria for this task: showcase *at least one* policy that rotates the cube around z axis for at least 10 seconds. The maximum grade is achievable also without the (bonus) subtask.

FULL TASK DOCUMENT AND CODE WILL BE RELEASED TONIGHT.
BE PATIENT