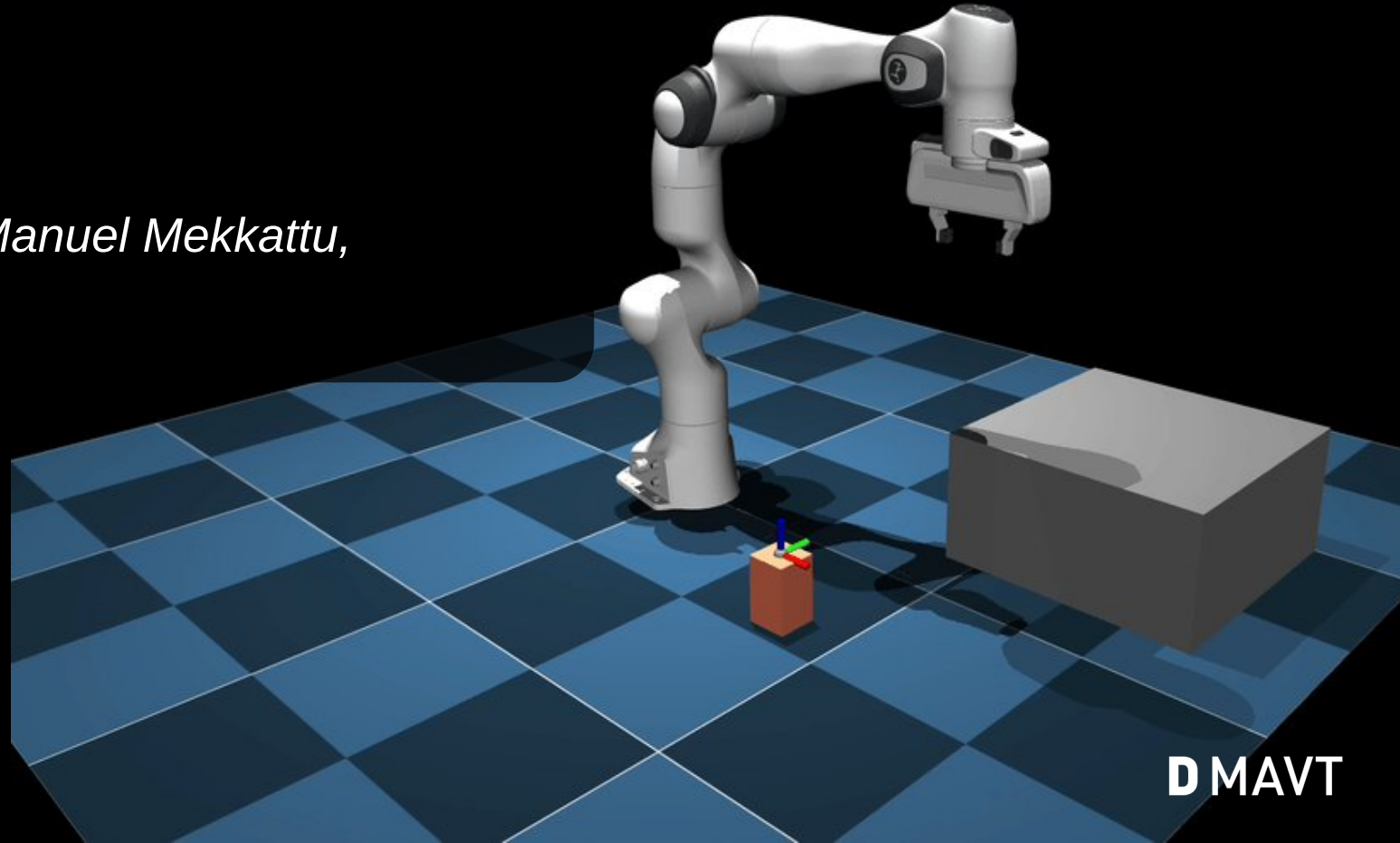




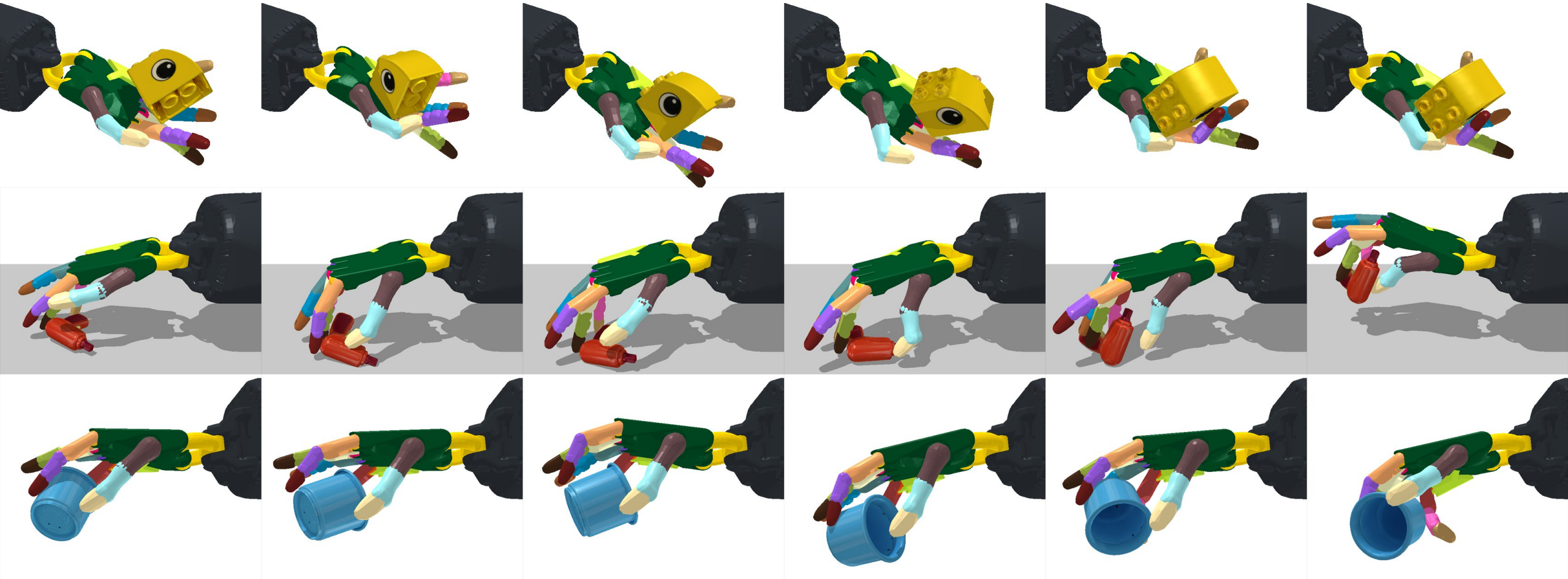
Methods and Challenges in Simulation

Focus Talk Unit 5

Mike Y. Michelis, Yasunori Toshimitsu, Manuel Mekkattu, Robert Katzschmann

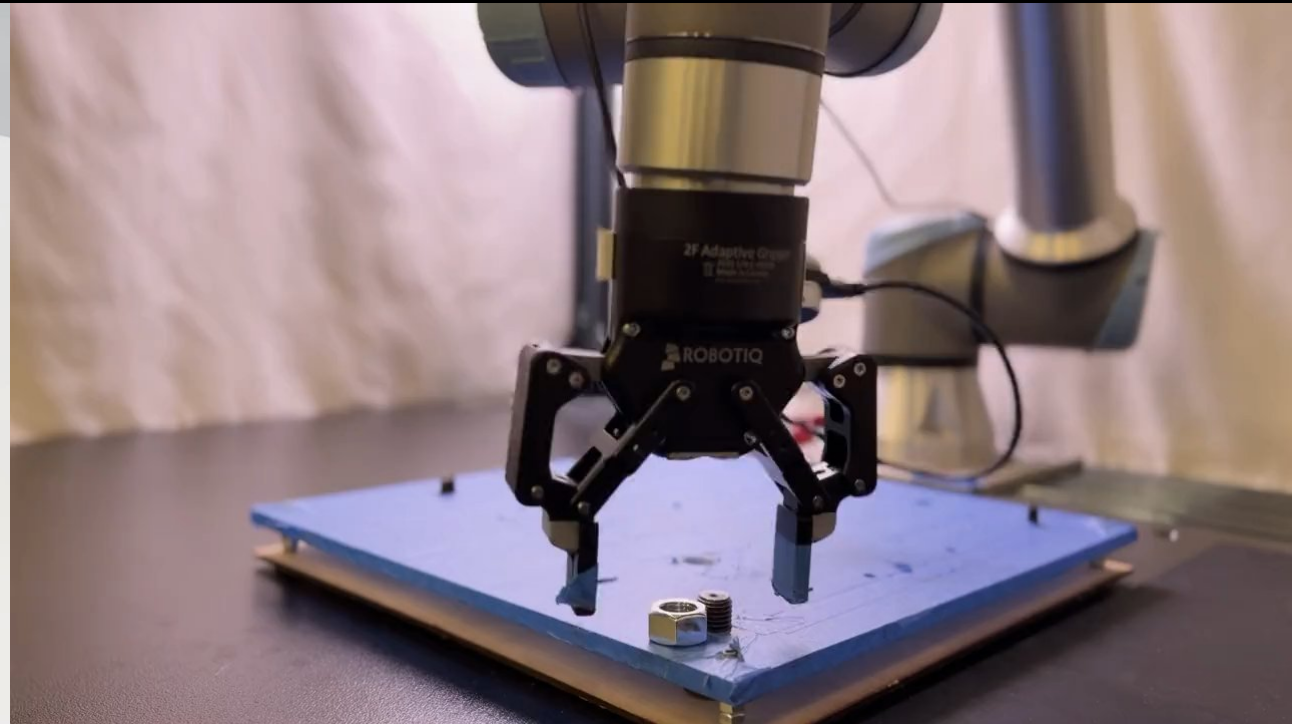
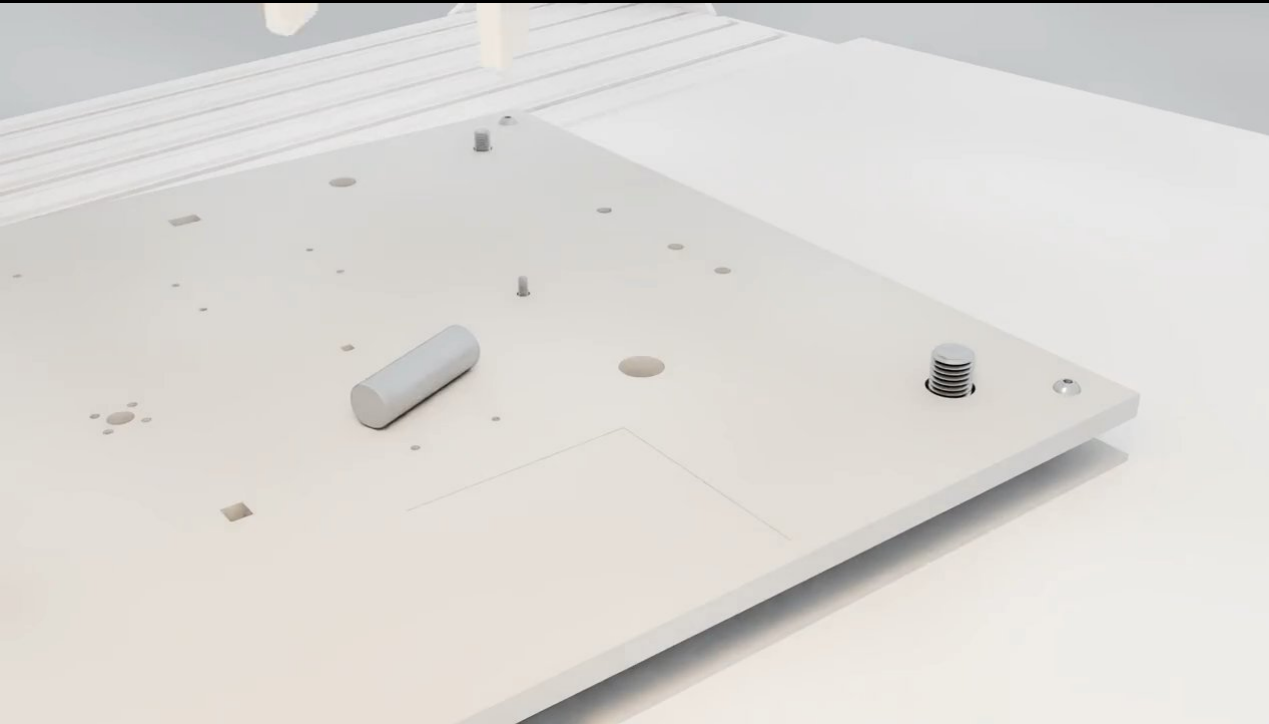


Simulating Dexterous Manipulation





Simulating Dexterous Manipulation





Why are simulators important?

Why are simulators important?



Safe and **fast** environment for testing robot controllers

Parallelized simulation for reinforcement learning (RL)

Model-based control based on simulators

Failure in real environments are expensive



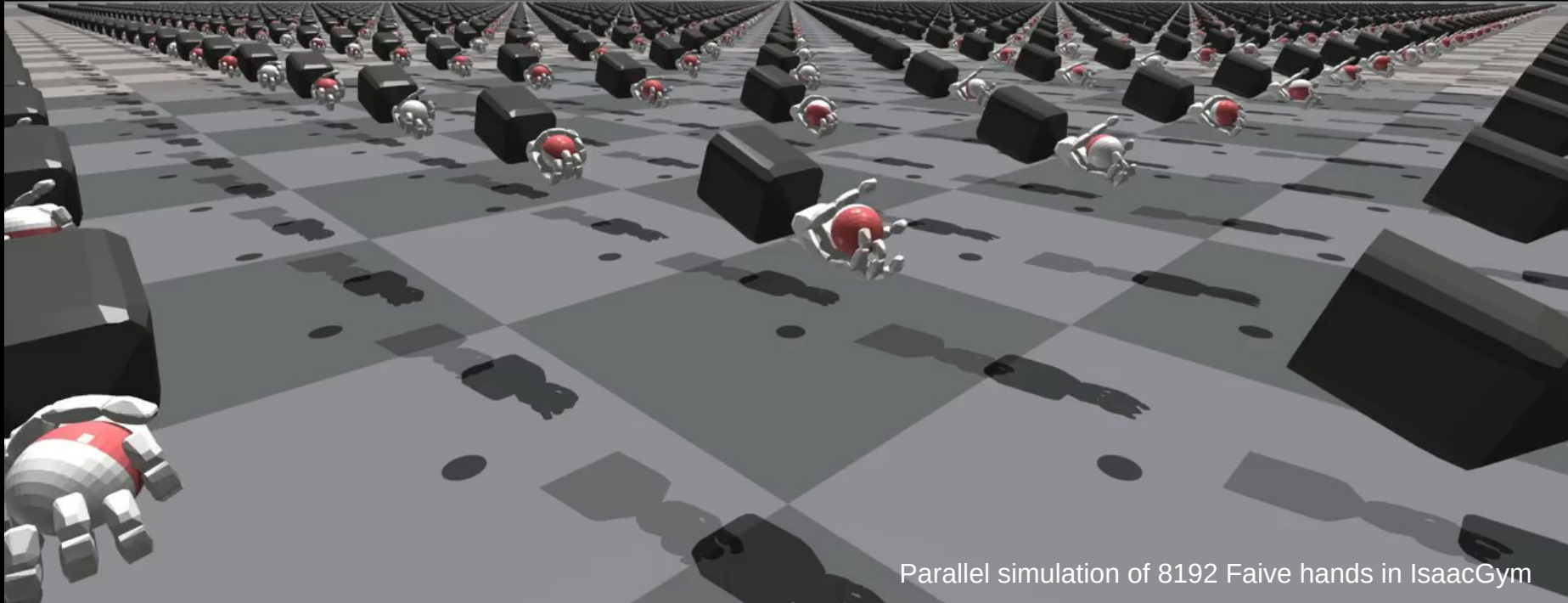
6:16:37 05/06/2015

Darpa Robotics Challenge (DRC): <https://www.darpa.mil/program/darpa-robotics-challenge>

Simulations are safe environments



Parallelized Simulation for RL

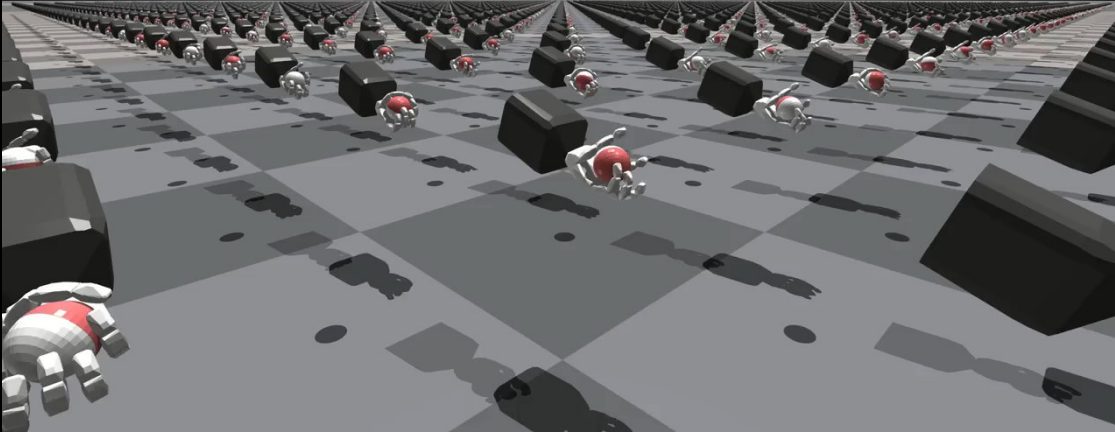


Parallelized Simulation

**Proximal Policy
Optimization**

Domain Randomization

Parallelized Simulation for RL



Parallel simulation of 8192 Faive hands in IsaacGym

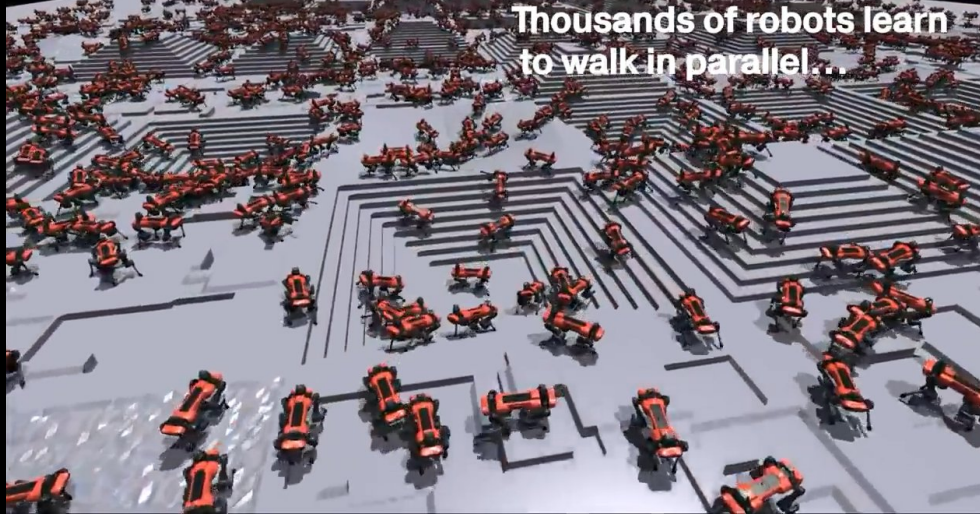
Parallelized Simulation

- Parallel robots on GPU^{1,2}
- Wide exploration of initial conditions

**Proximal Policy
Optimization**

Domain Randomization

Parallelized Simulation for RL



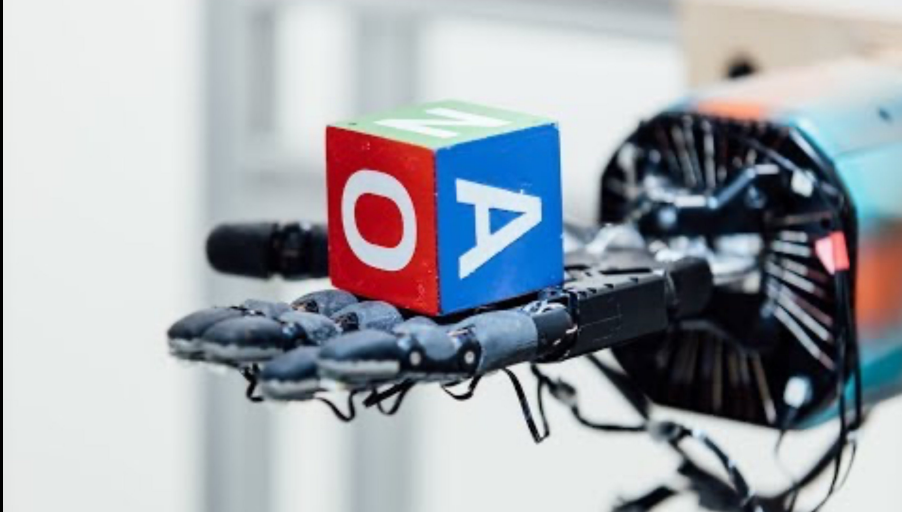
Proximal Policy Optimization (PPO)

- Reinforcement Learning
- Scales well to parallel environments²

Parallelized Simulation

Domain Randomization

Parallelized Simulation for RL



Domain Randomization

- Randomize physics
- Add noise to observations
- Make it robust for physical deployment

Parallelized Simulation

**Proximal Policy
Optimization**

Parallelized Simulation for RL



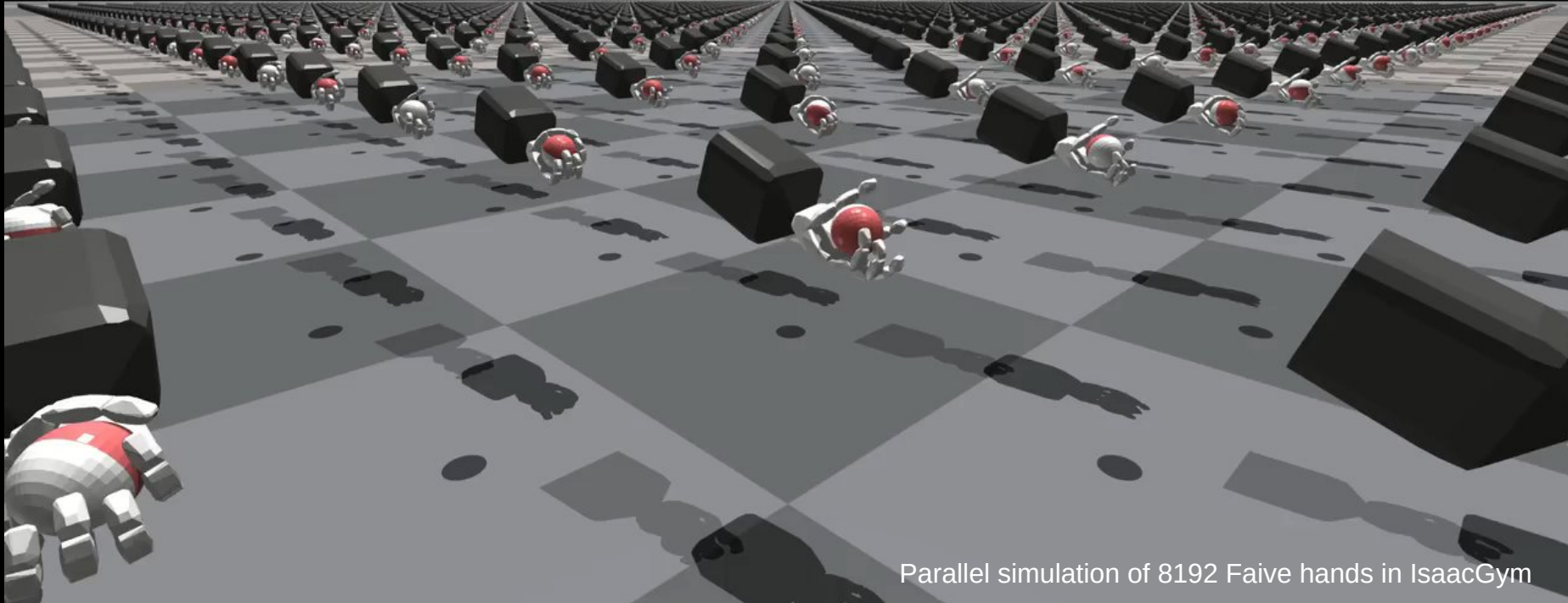
Domain Randomization

- Randomize physics
- Add noise to observations
- Make it robust for physical deployment

Parallelized Simulation

**Proximal Policy
Optimization**

Parallelized Simulation for RL



Parallelized Simulation

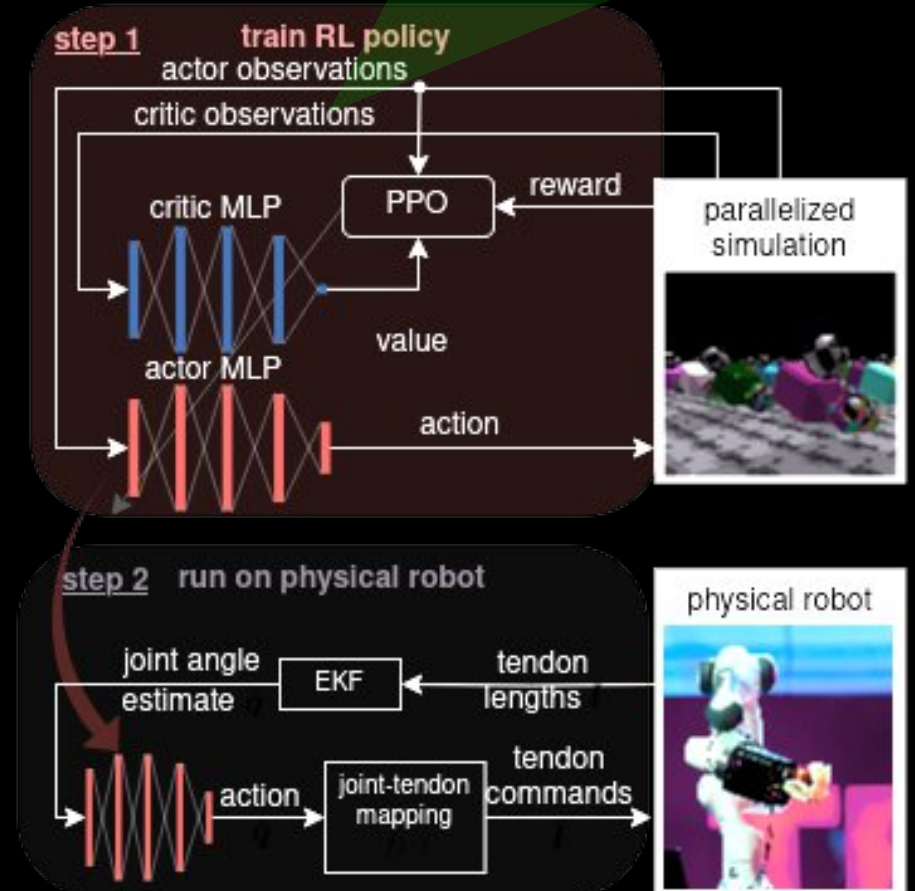
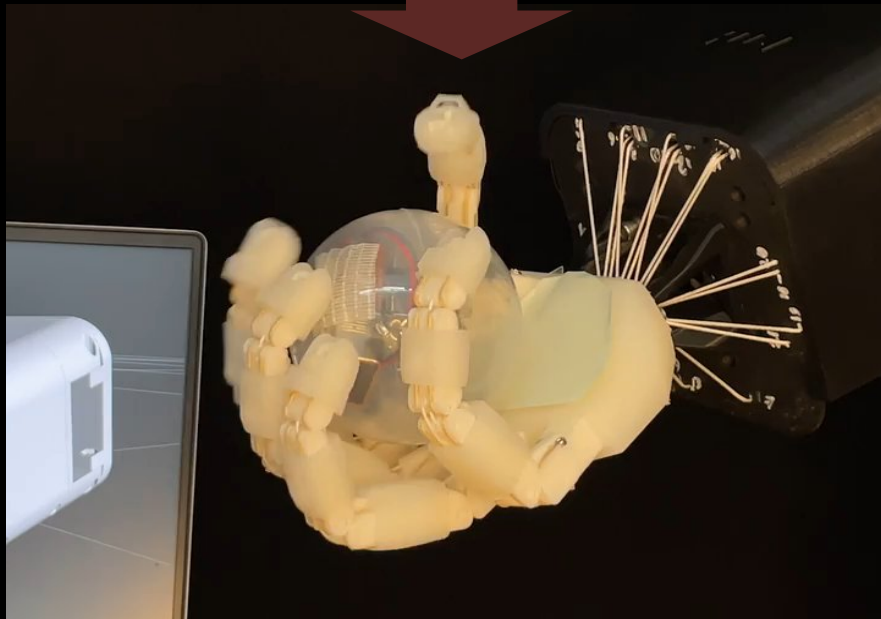
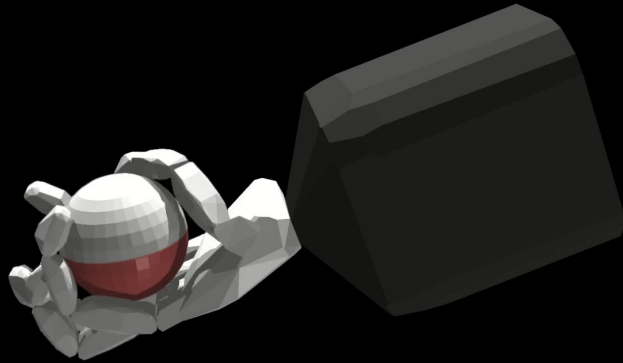
**Proximal Policy
Optimization**

Domain Randomization

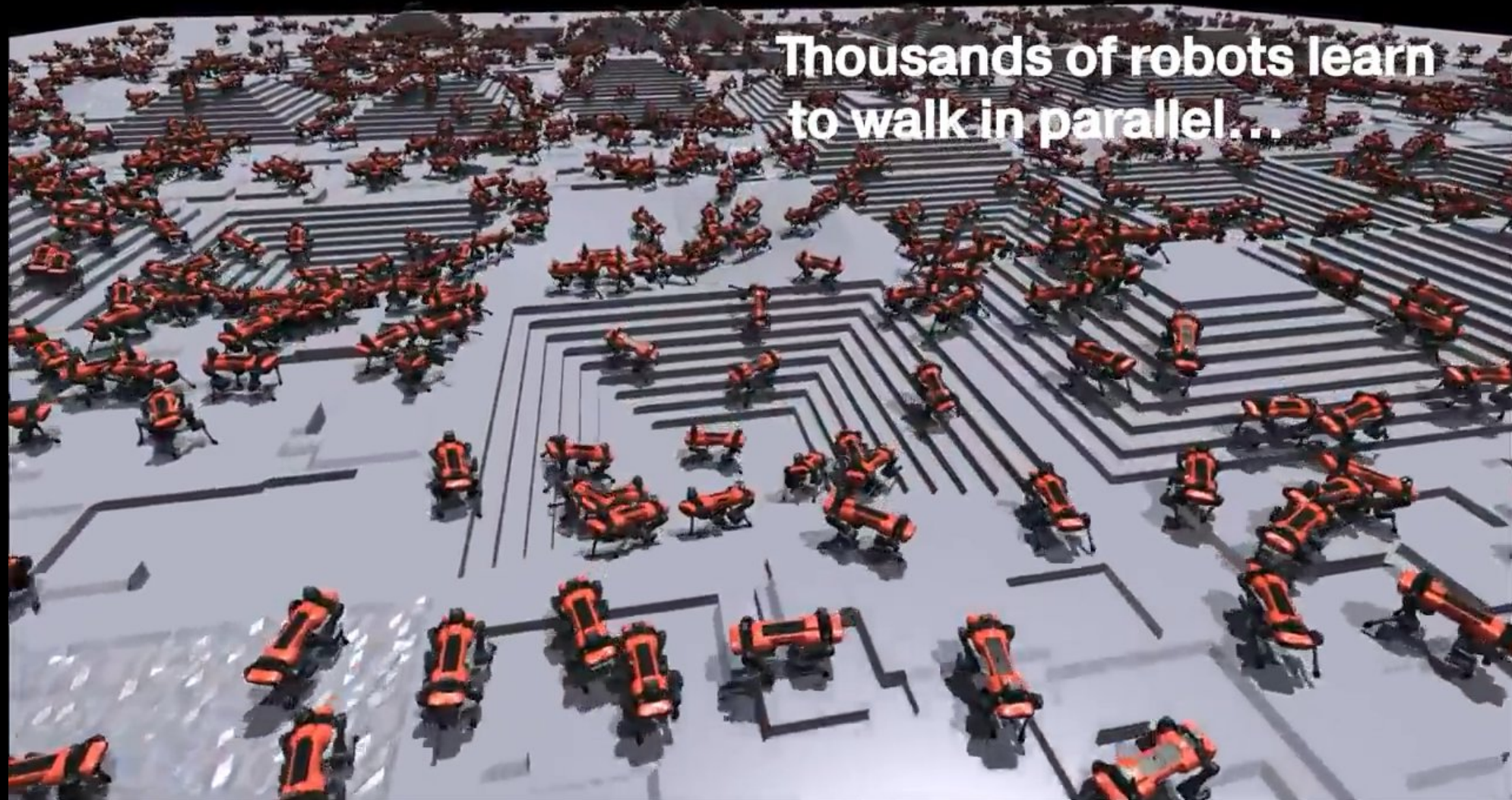
Sim-to-Real for Dexterous Manipulation



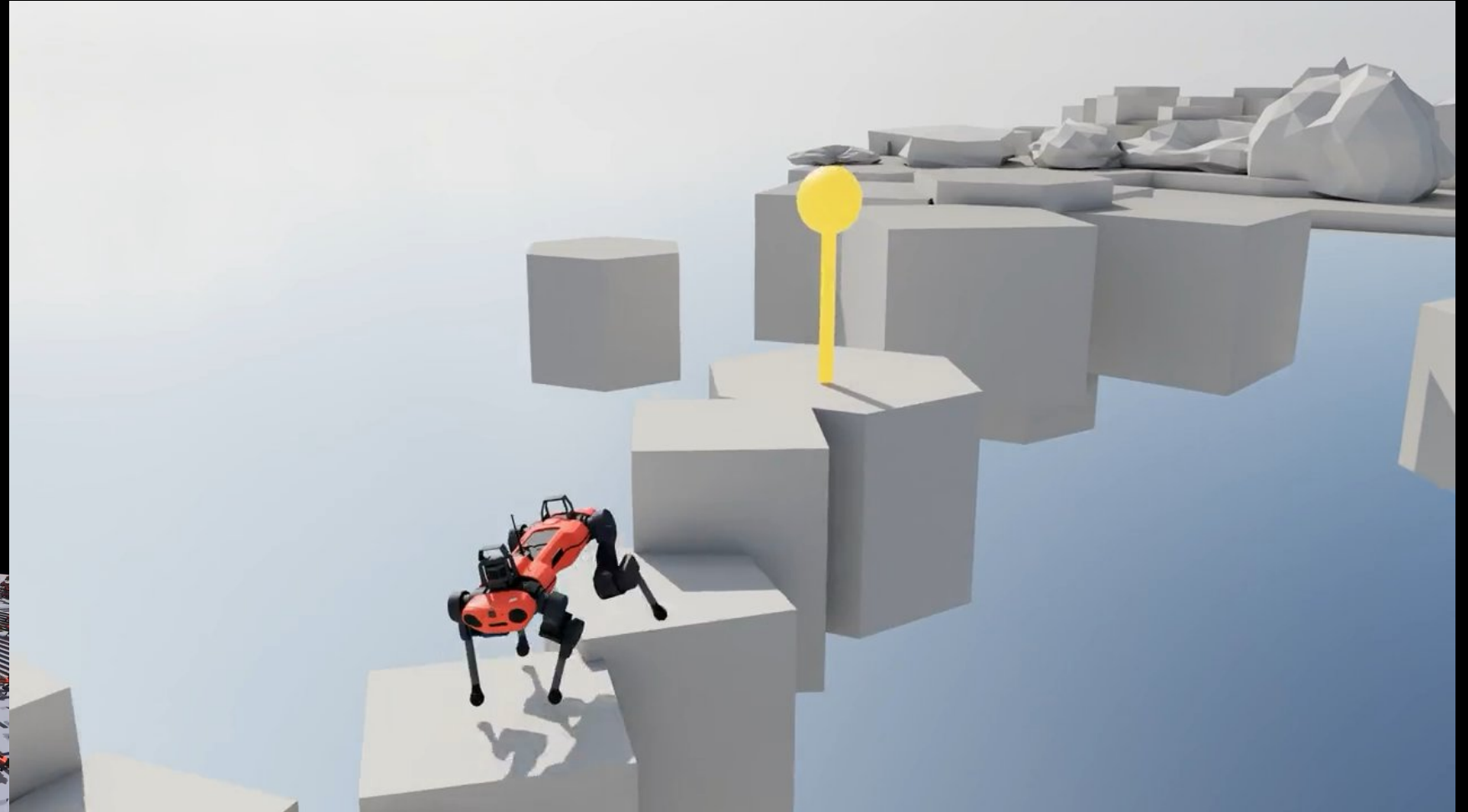
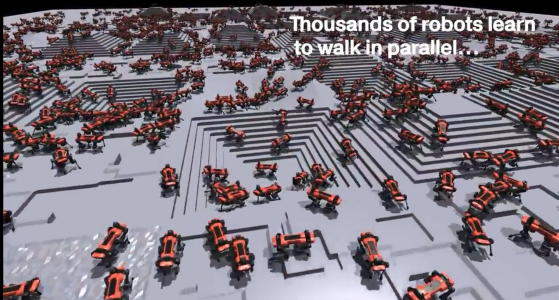
1 hour of training at 70,000 fps $\approx$ 2 months of simulated time



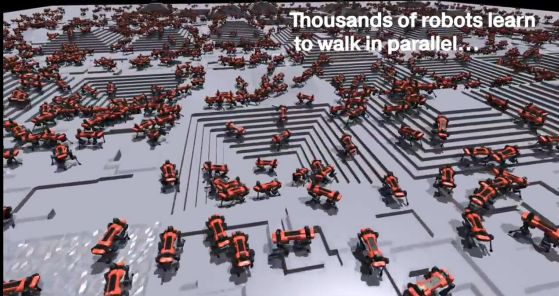
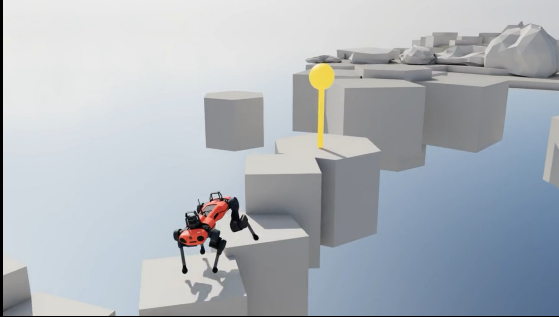
Massively parallel RL across domains



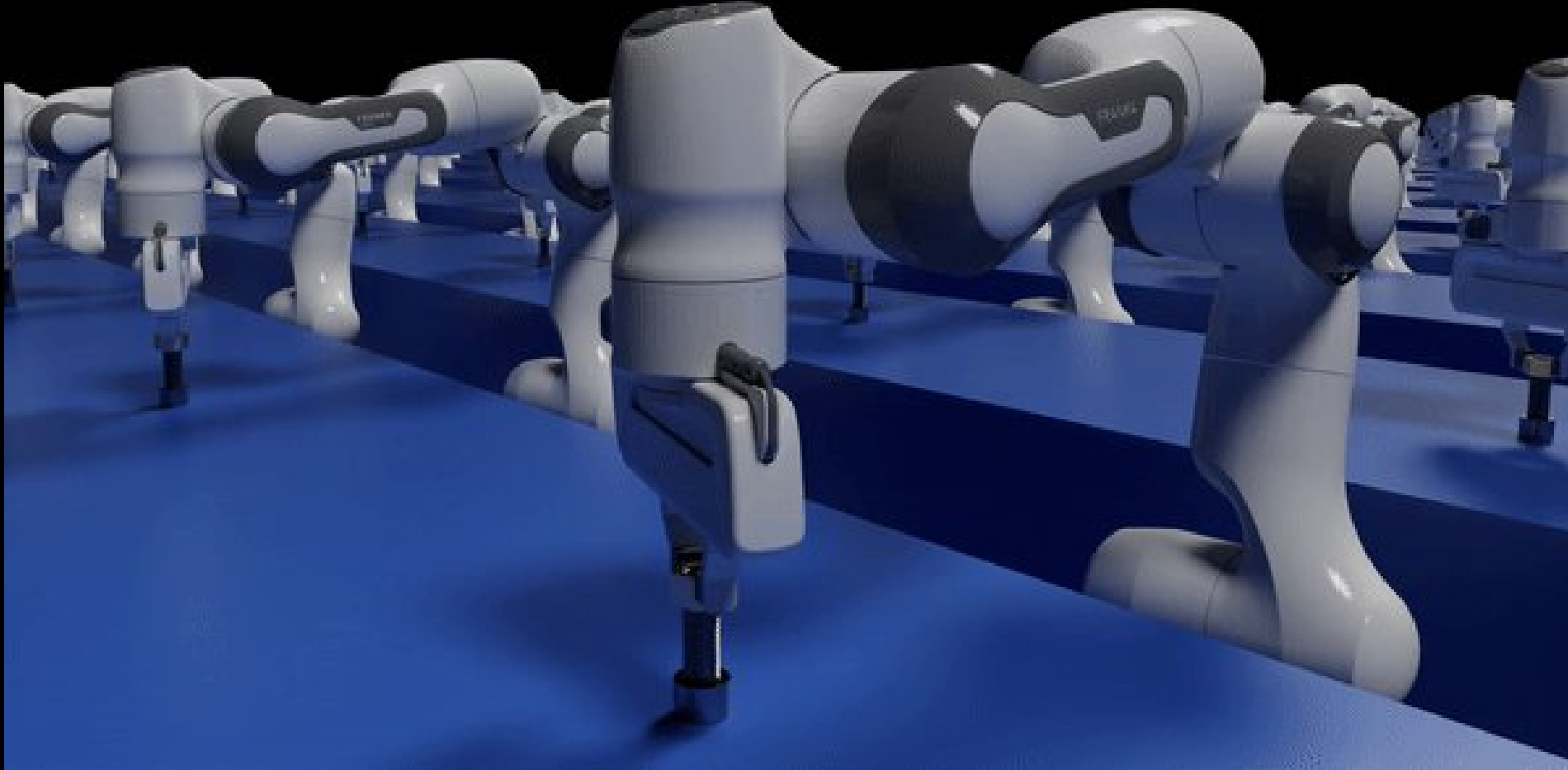
Massively parallel RL across domains



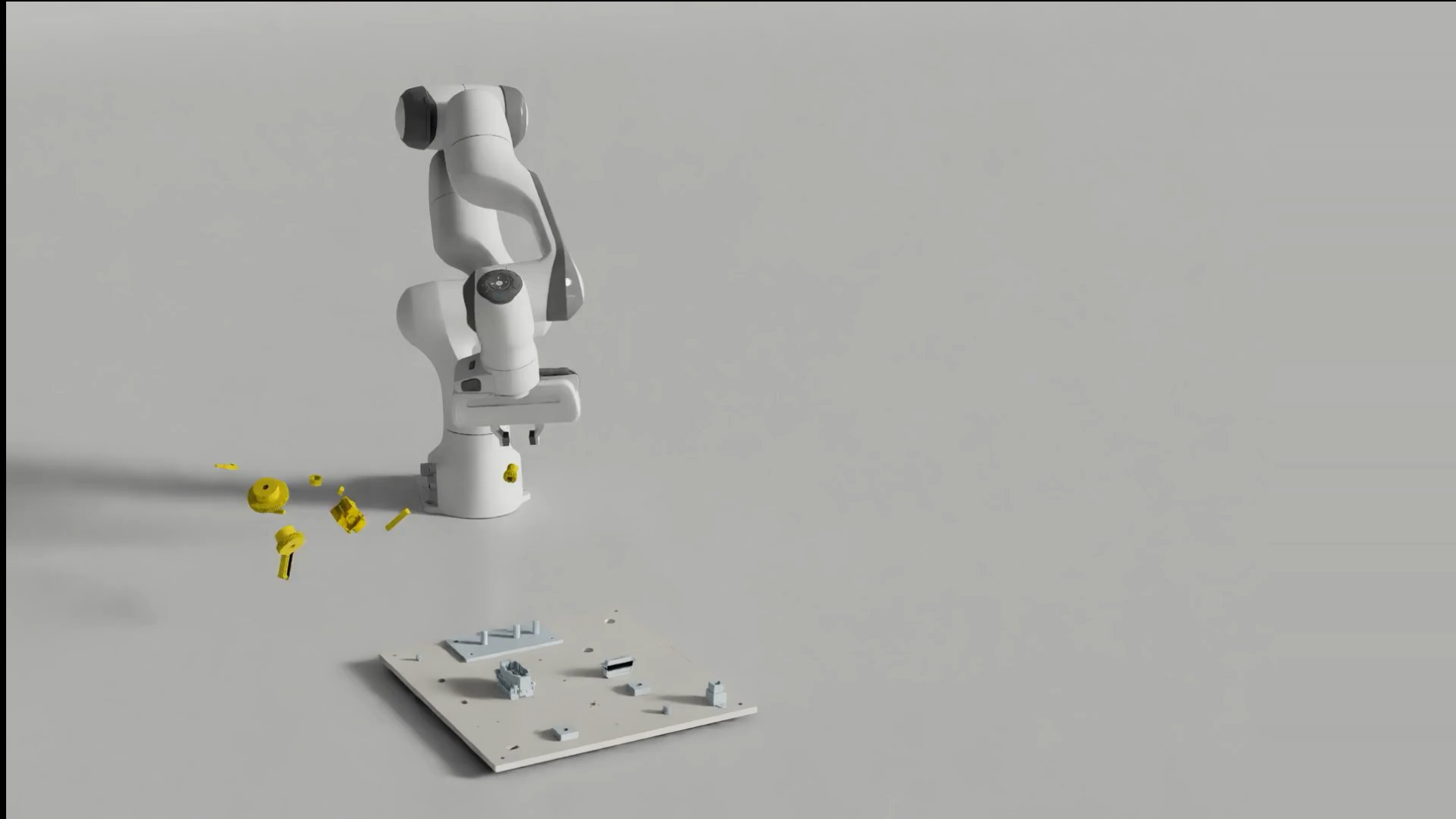
Massively parallel RL across domains



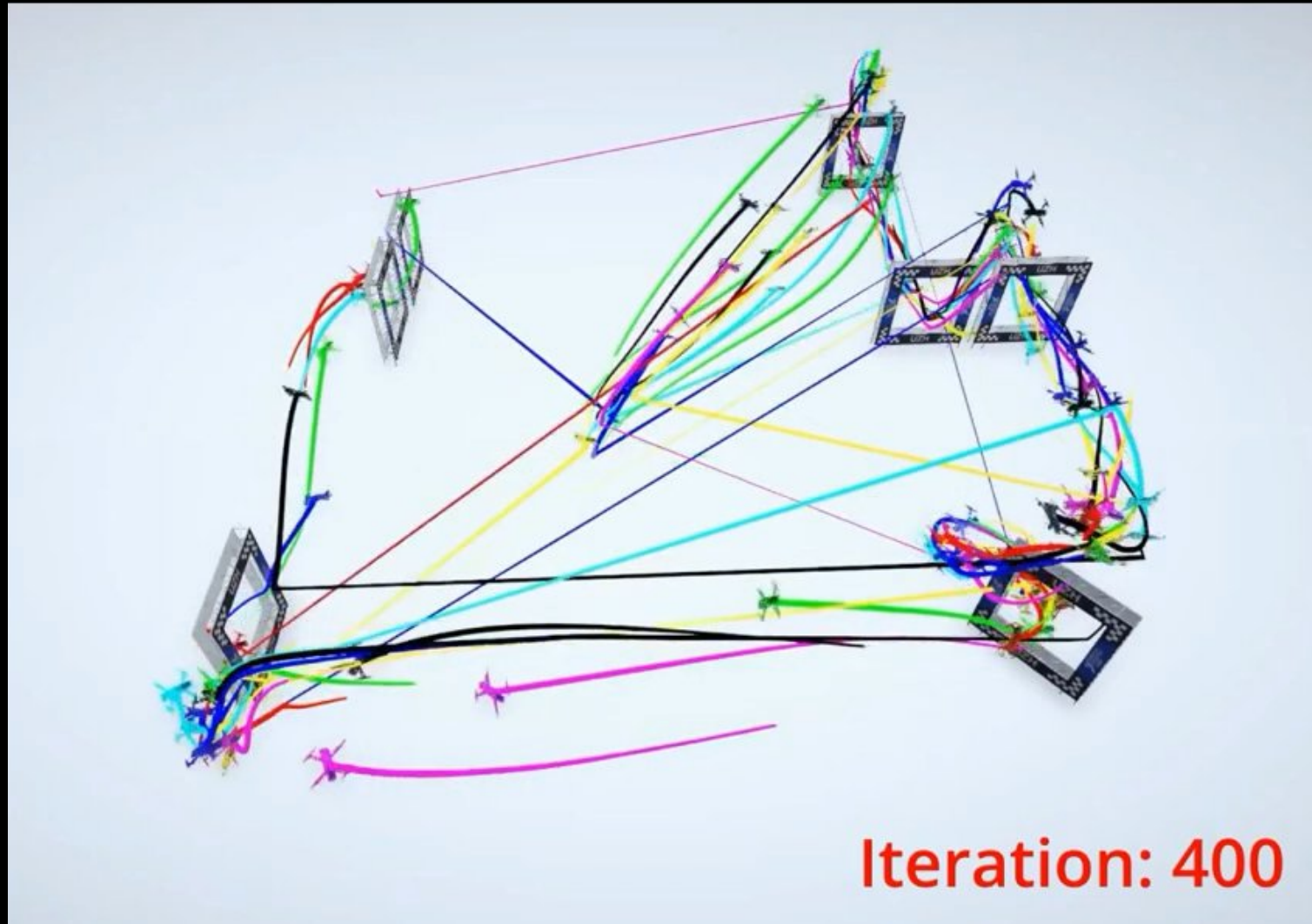
Massively parallel RL across domains



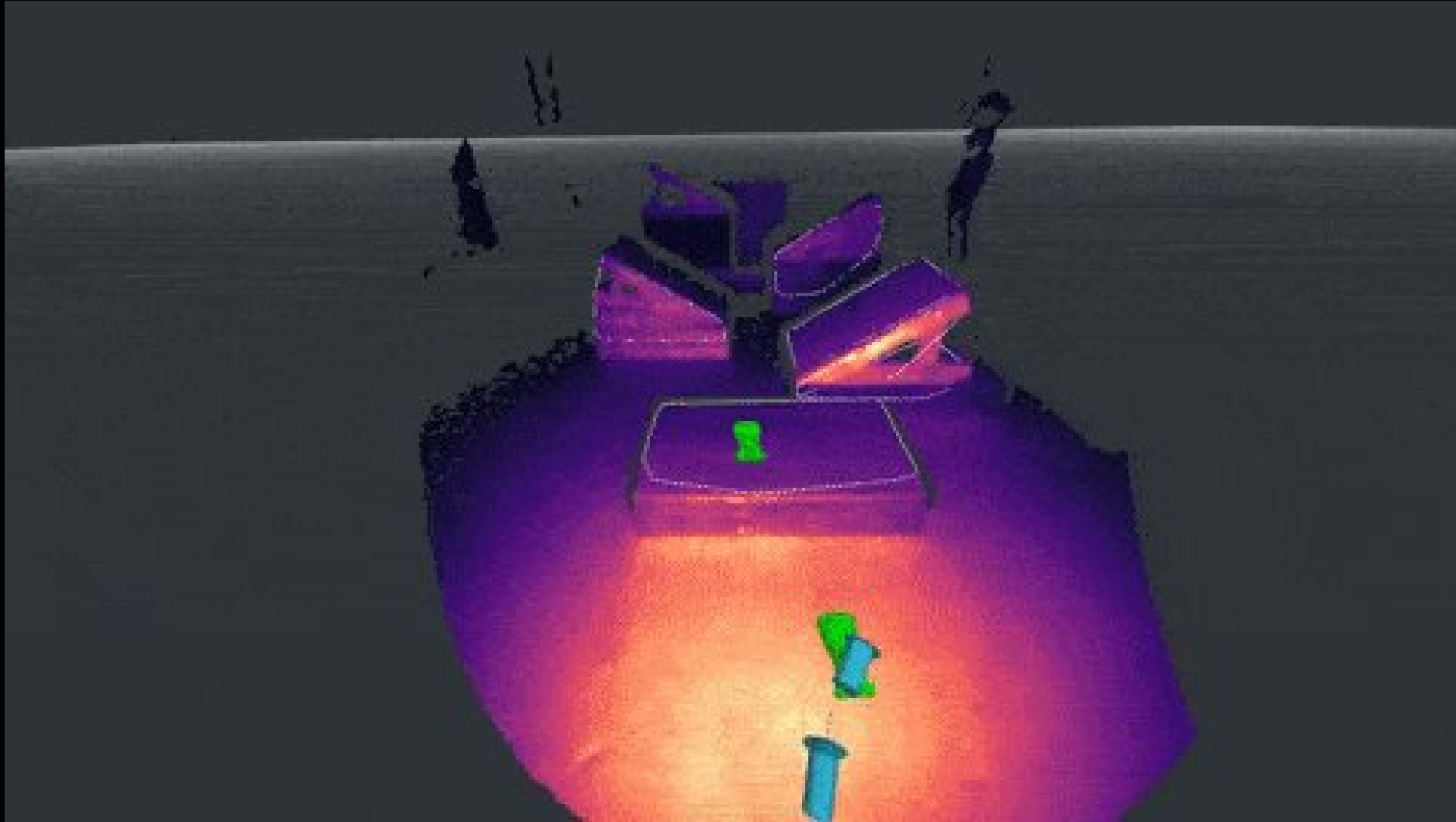
Massively parallel RL across domains



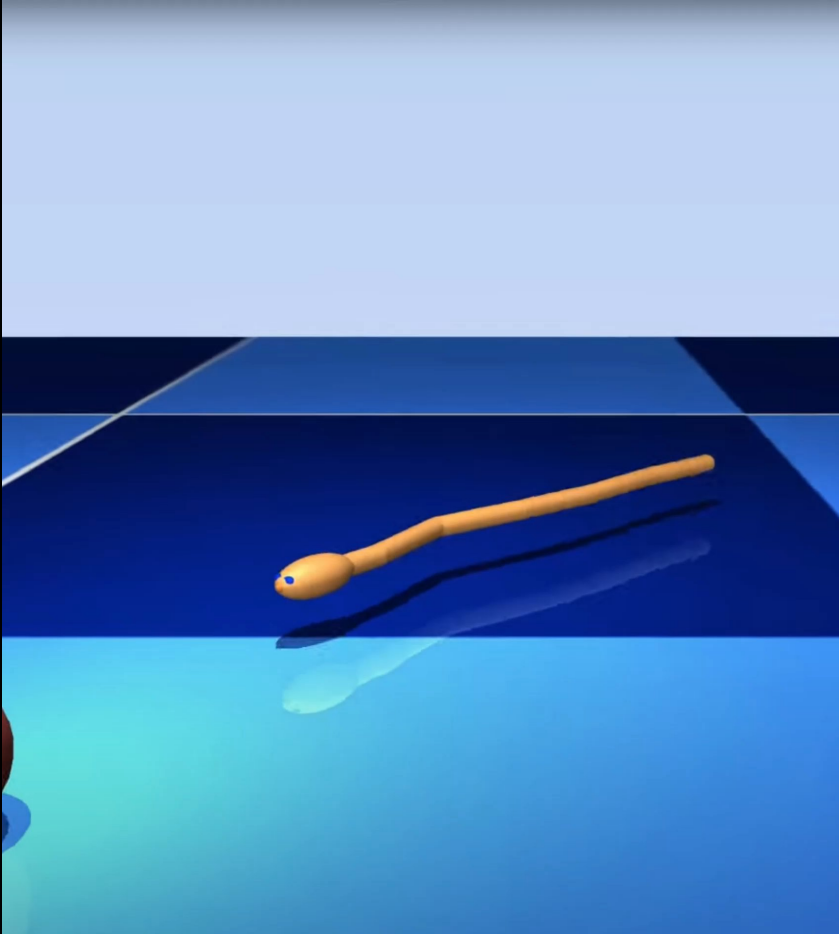
Massively parallel RL across domains



Online Rigid Body Sim for BD Atlas MPC



MPC on Simulated MuJoCo Model



MPC: model predictive control

- Define a cost function to minimize, encoding the desired task
- Iteratively optimize “best” control inputs per step

MuJoCo-MPC uses MuJoCo as the **model** in MPC
→ efficient and accurate model-based control

https://github.com/google-deepmind/mujoco_mpc



Key idea when simulating robots:
Simplification

Why simplify?



Efficiency

trade off some accuracy for speed

For example...

RL → efficient simulation enables faster training

MPC → faster exploration enables higher control frequency, longer control horizon

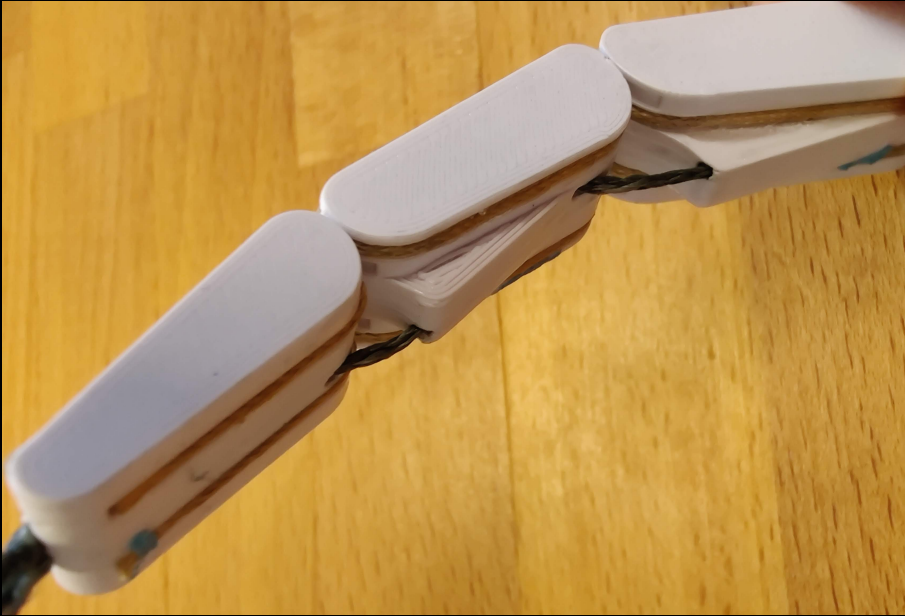
Modularity

If the real robot has a good low-level controller, the simulator can use high-level control inputs

For example...

The tendons of the Faive Hand are not (currently) replicated in the simulated model, and it accepts joint-level commands which are easier to learn

Simplification example #1: rolling contact joints

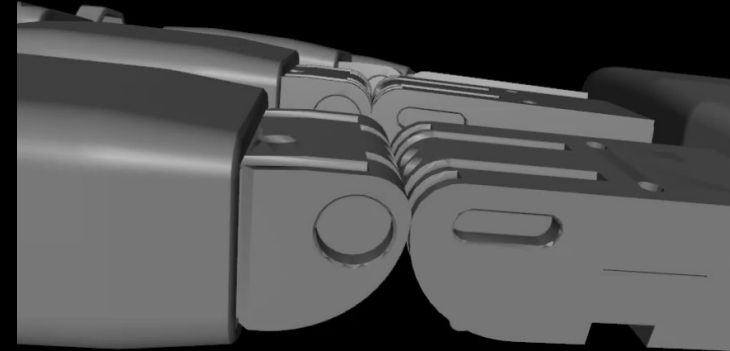


Real robot

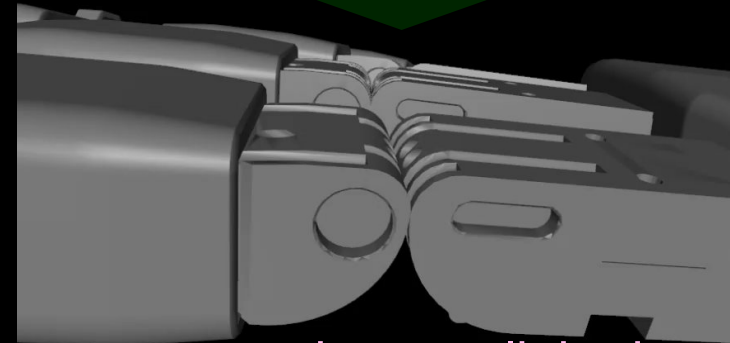
Contact between cylindrical surfaces

Ligaments ensure that they don't slide or move apart

pause



Joints not linked



Joints are linked

thumb_mp2d	0
root2index_p	0
root2index_p	0
index_pp2m	0
index_pp2m	0
index_mp2d	0
index_mp2d	0
root2middle_	0
root2middle_	0
middle_pp2	0
middle_pp2	0
middle_mp2	0

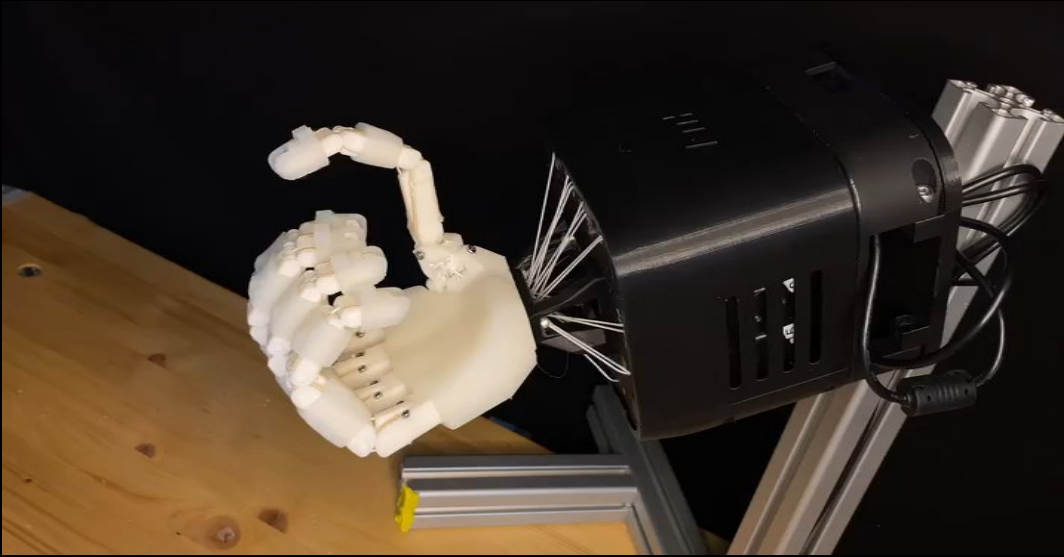
Clear all	
root2thumb_	0
thumb_base	0
thumb_pp2m	0
root2index_p	0
index_pp2m	0
root2middle_	0
middle_pp2	0
root2ring_pp	0
ring_pp2mp	0
root2pinky_p	0
pinky_pp2m	0

Simulated robot

Two hinge joints make up a single rolling contact joint

Constrained to roll together when the joint is actuated

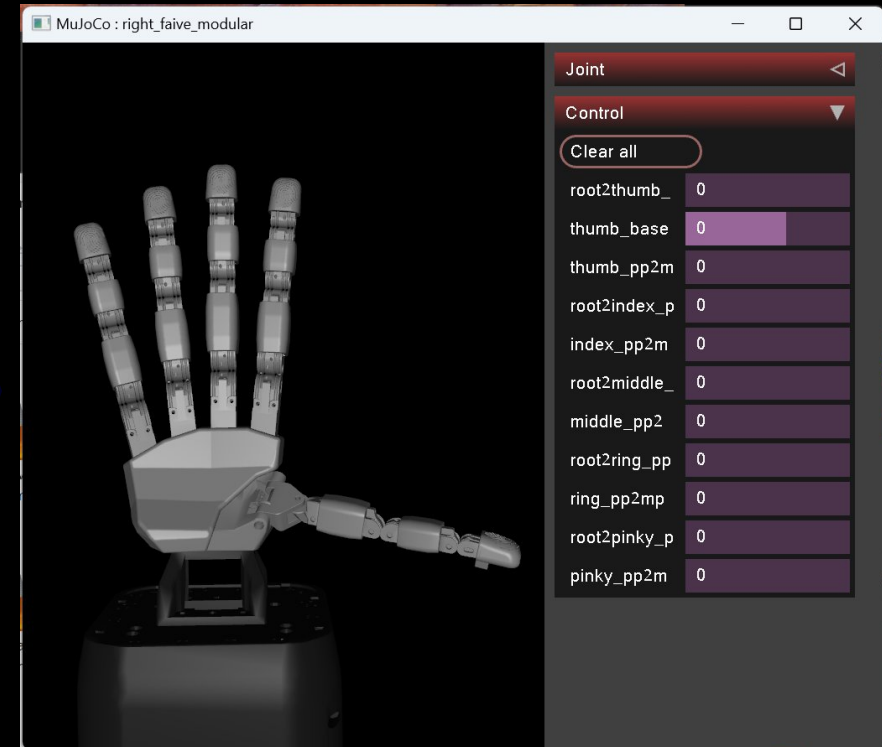
Simplification example #2: tendon-driven actuation



Real robot

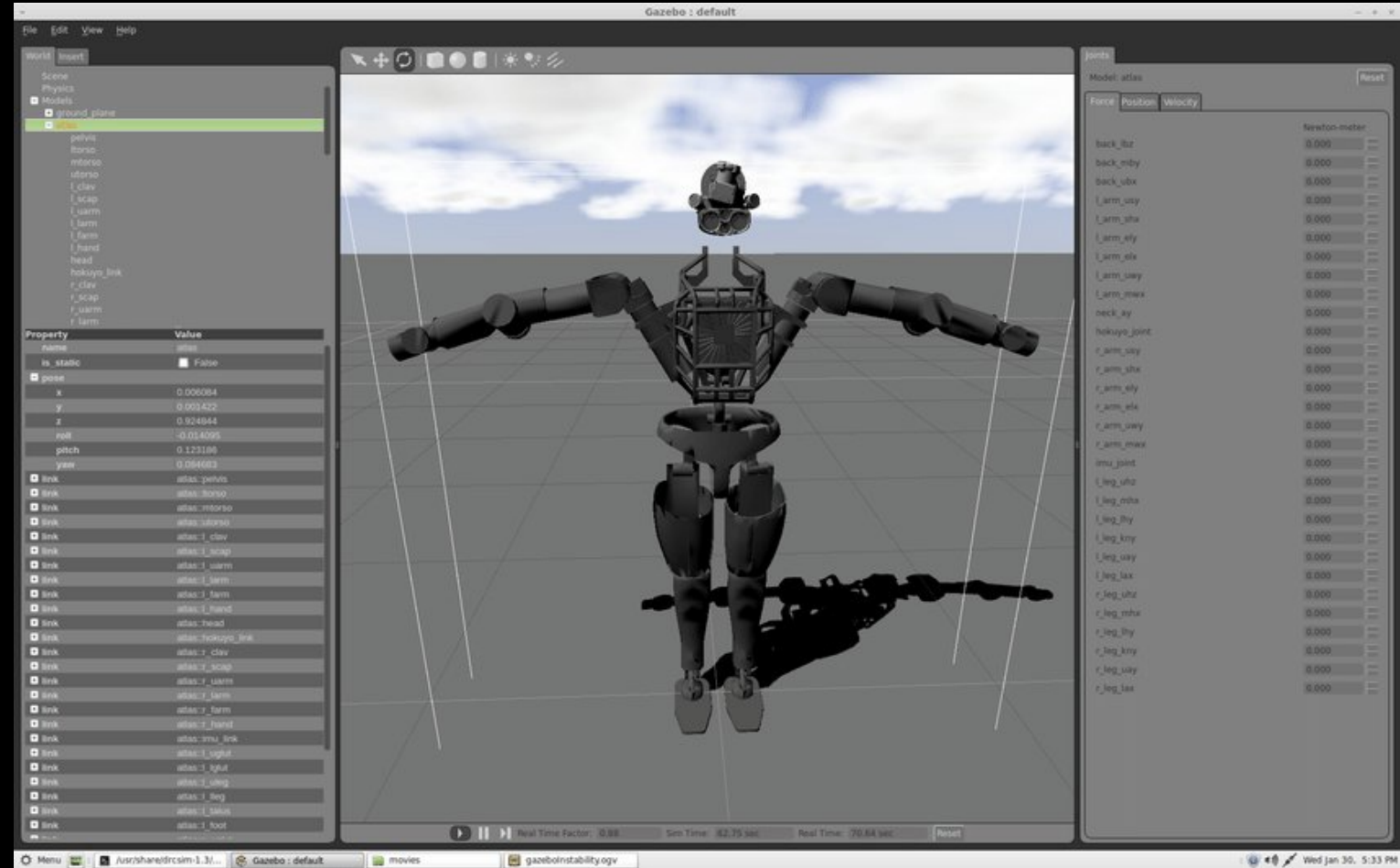
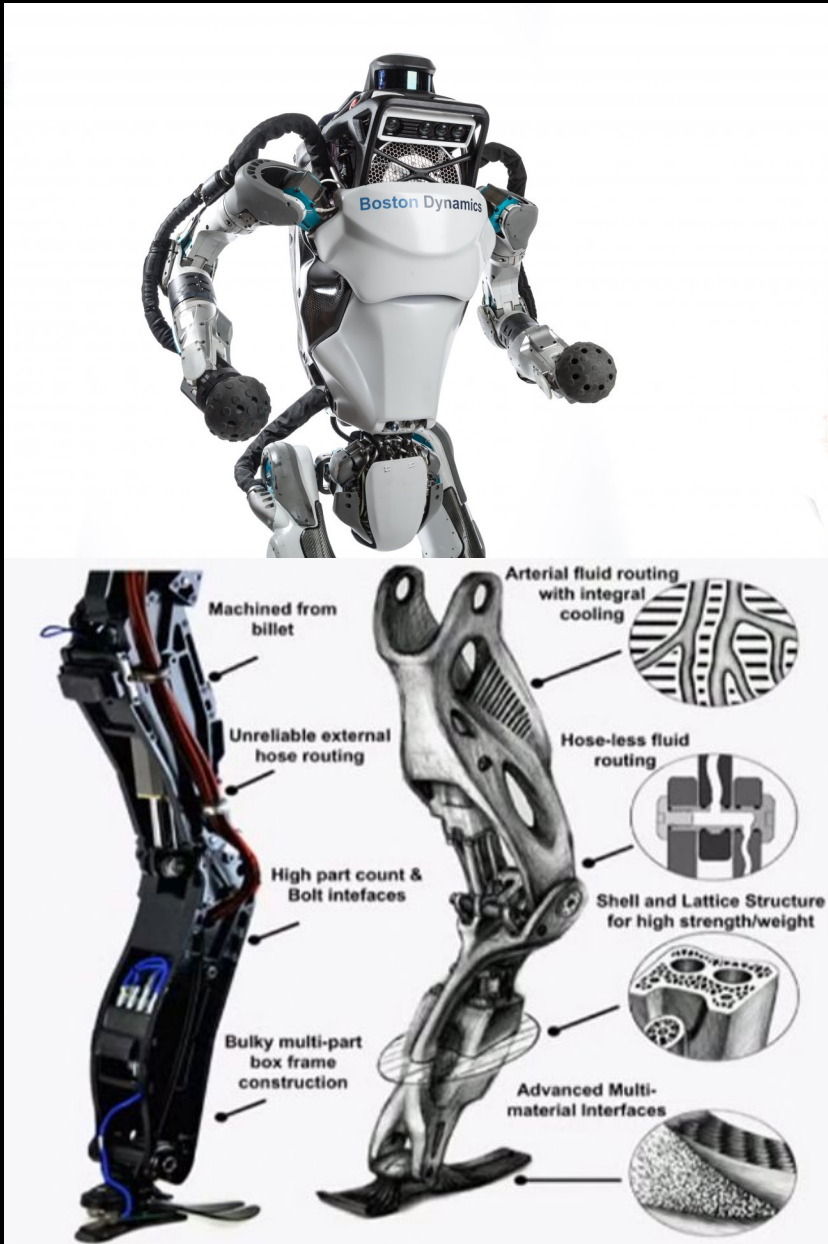
16 servo motors drive the tendons to actuate the hand

Low-level controller converts joint angles → tendon lengths → servo motor angles and sends it to Dynamixel motors



Simulated robot

The musculoskeletal system is ignored, and the robot is modelled purely as a joint axis-driven robot





Take-Away

Simulate and fail often to simplify later real-world experiments

Summary of why simulators are needed



Safe and **fast** environment for testing robot controllers

Parallelized simulation for reinforcement learning (RL)

Model-based control based on simulators

What simulators are out there?



Important questions for simulators:

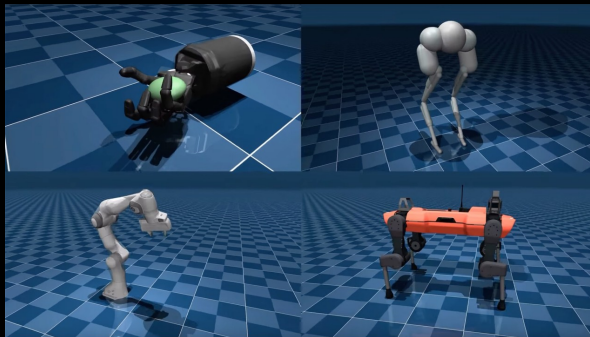
Stability and accuracy? · GPU parallelization (i.e. speed)?
Open source? · Extensive documentation / large community?

“Team Google”



Google DeepMind

MuJoCo



SoftRobotics
Laboratory



(but not really)

“Team NVIDIA”



NVIDIA

Isaac Sim / NVIDIA Omniverse



MuJoCo (Multi-Joint dynamics with Contact)

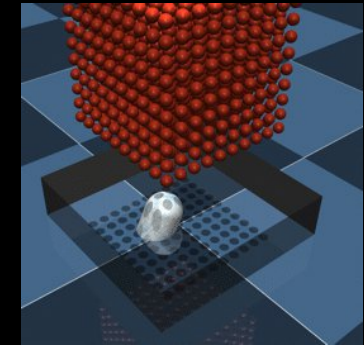
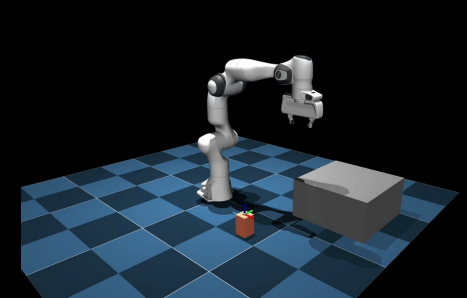
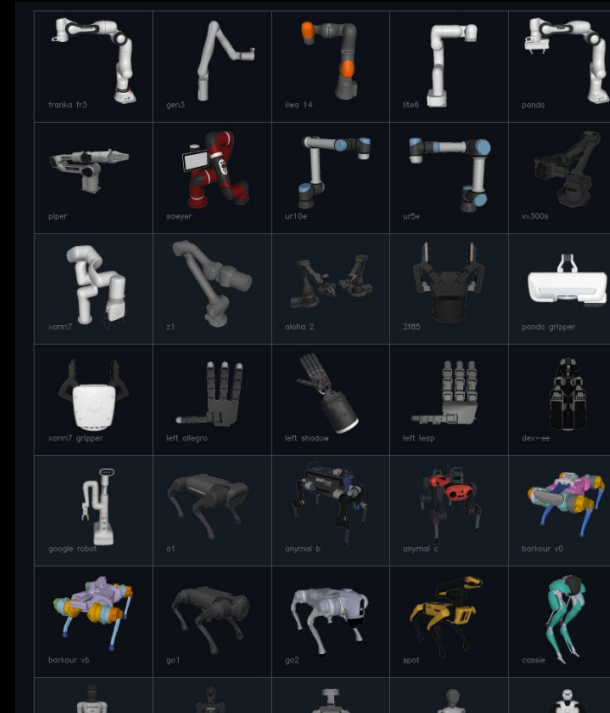


Origins and features

MuJoCo XLA (MJX)

MuJoCo Warp (MJWarp)

Collection of high-quality MuJoCo assets: https://github.com/google-deepmind/mujoco_menagerie



MuJoCo

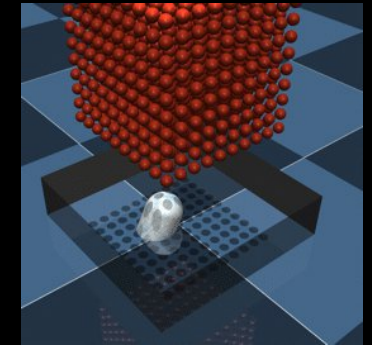
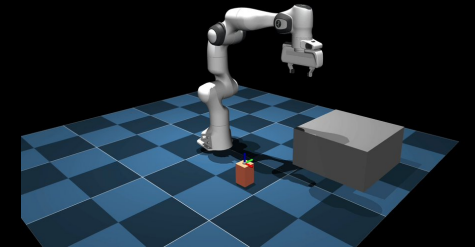
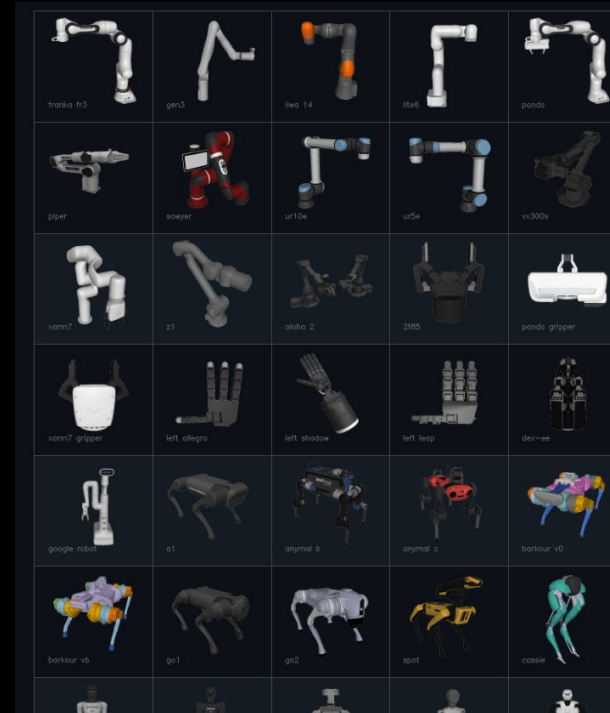
MuJoCo (Multi-Joint dynamics with Contact)



Origins and features

- Originally developed by Emo Todorov et al. (just 3 people!)
- Optimization-based contact solver for efficiency, accuracy, and stability
- Acquired by Google DeepMind in 2021, open sourced 7 months later
- Supports URDF or MJCF (MuJoCo format)
- Originally CPU-only, but...

Collection of high-quality MuJoCo assets: https://github.com/google-deepmind/mujoco_menagerie



MuJoCo

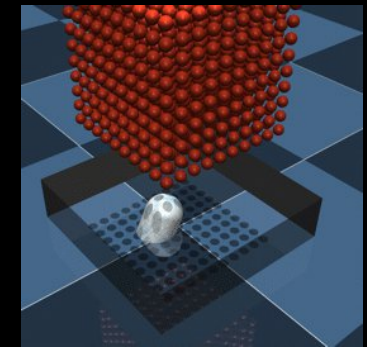
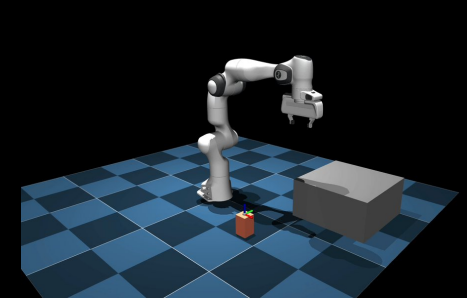
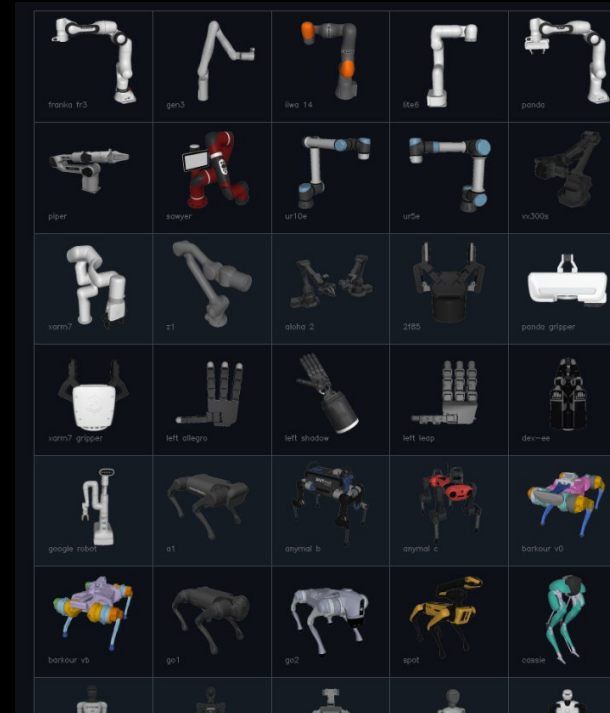
MuJoCo (Multi-Joint dynamics with Contact)



MuJoCo XLA (MJX)

- JAX/XLA rewrite of MuJoCo
- runs on CPU, GPU, or TPU
- Doesn't support some features ■■

Collection of high-quality MuJoCo assets: https://github.com/google-deepmind/mujoco_menagerie



MuJoCo

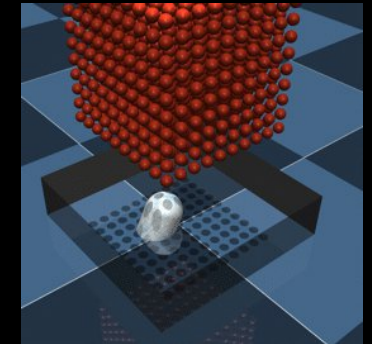
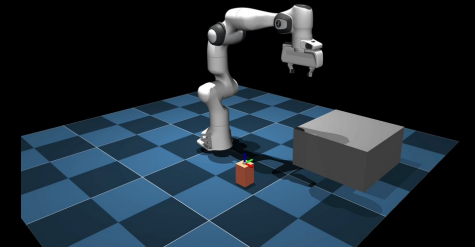
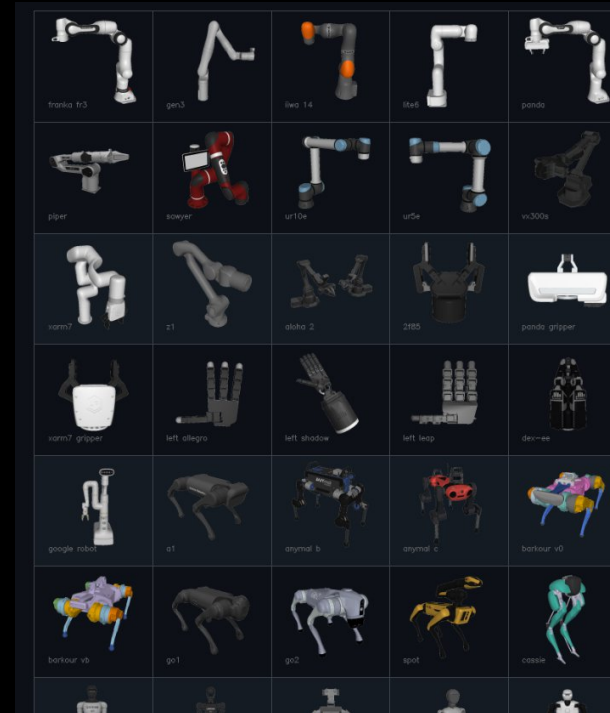
MuJoCo (Multi-Joint dynamics with Contact)



MuJoCo Warp (MJWarp)

- Team up with NVIDIA
- Now in Beta
- GPU-optimized reimplementation to run on NVIDIA hardware
- Should support more MuJoCo features than MJX
- Will be integrated into MJX in future

Collection of high-quality MuJoCo assets: https://github.com/google-deepmind/mujoco_menagerie



MuJoCo

MuJoCo (Multi-Joint dynamics with Contact)

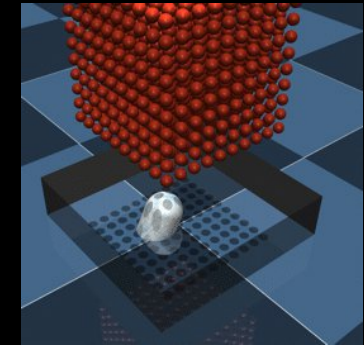
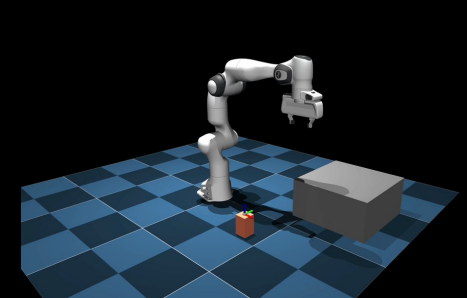
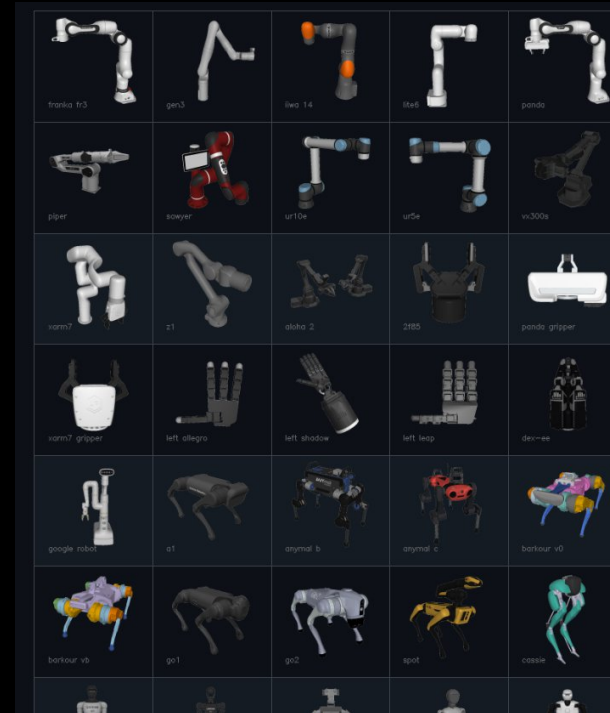


Origins and features

MuJoCo XLA (MJX)

MuJoCo Warp (MJWarp)

Collection of high-quality MuJoCo assets: https://github.com/google-deepmind/mujoco_menagerie



MuJoCo

NVIDIA Isaac Sim / Isaac Lab



Isaac Sim

Isaac Lab

Lab is compatible with **Newton**

MuJoCo



In our hands-on part, we will implement dexterous RL on the ORCA hand using **faive_lab**, an extension of Isaac Lab!

NVIDIA Isaac Sim / Isaac Lab



Isaac Sim

- simulator using PhysX for the physics engine
- Apache-2.0 open source (building/using it requires other NVIDIA licensed components)



In our hands-on part, we will implement dexterous RL on the ORCA hand using **faive_lab**, an extension of Isaac Lab!

MuJoCo



NVIDIA Isaac Sim / Isaac Lab



Isaac Lab

- Wraps Isaac Sim, to build RL / IL workflows
- Only runs on NVIDIA's GPUs



In our hands-on part, we will implement dexterous RL on the ORCA hand using **faive_lab**, an extension of Isaac Lab!

MuJoCo



NVIDIA Isaac Sim / Isaac Lab



Lab is compatible with **Newton**

- open-source GPU physics engine
- MuJoCo-Warp solver



In our hands-on part, we will implement dexterous RL on the ORCA hand using **faive_lab**, an extension of Isaac Lab!

MuJoCo



NVIDIA Isaac Sim / Isaac Lab



Isaac Sim

Isaac Lab

Lab is compatible with **Newton**

MuJoCo



In our hands-on part, we will implement dexterous RL on the ORCA hand using **faive_lab**, an extension of Isaac Lab!

Other Simulators Coming Up



Other Simulators Coming Up

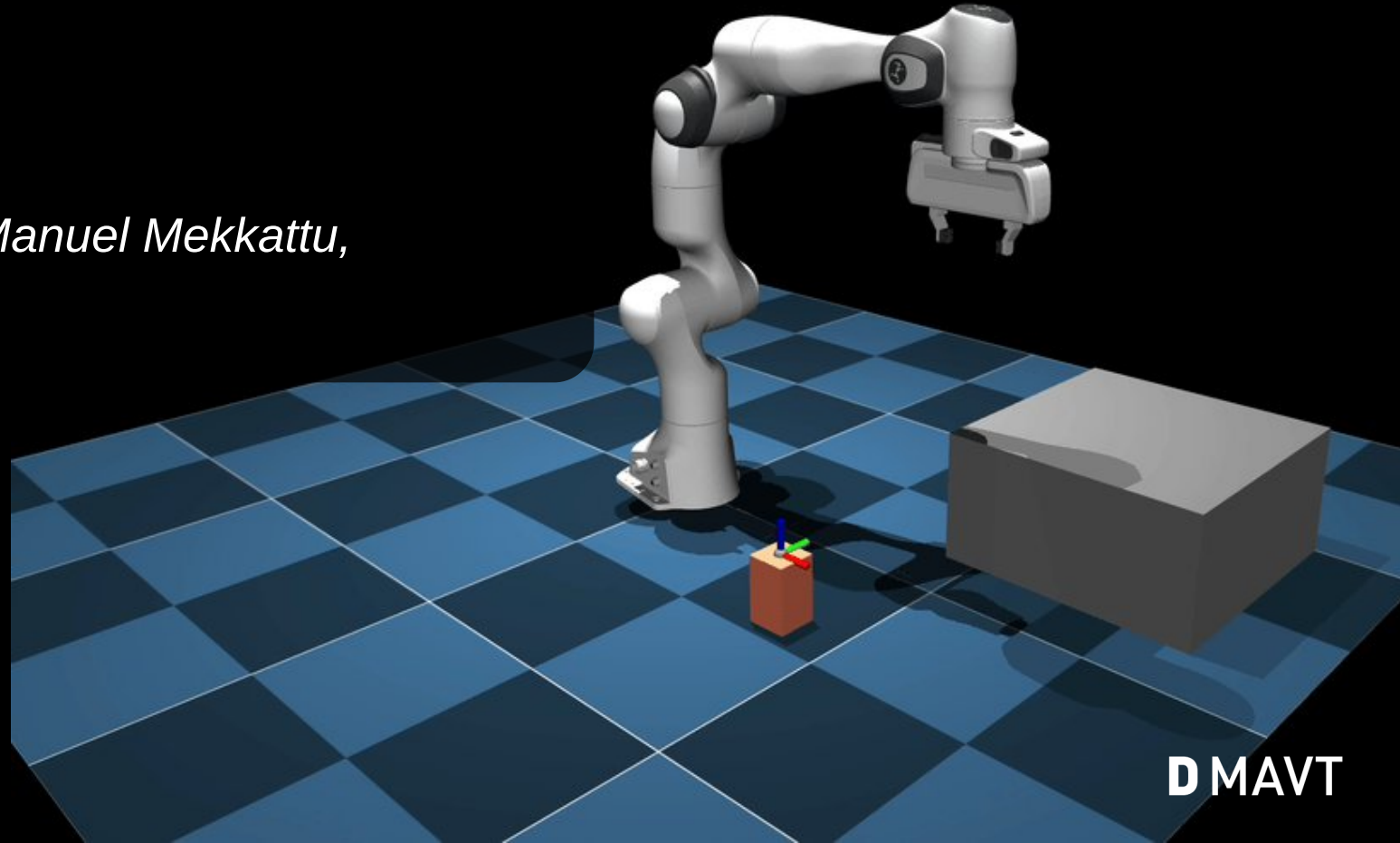




Methods and Challenges in Simulation

Focus Talk Unit 5

Mike Y. Michelis, Yasunori Toshimitsu, Manuel Mekkattu, Robert Katzschmann



Feedback to the Tutorial on Simulation – Part 1



- **Minute 6:21:** *the background music is too loud*
 - We will fix it in the next revision of this video.
- **Minute 11:27:** *What kind of limitations does URDF impose on the robot representation ? Are they really important in the case of soft robotics or is it just for convenience that we use MuJoCo's format in this course ?*

Tutorial needs updating

 - Practically I find it more viewable since MJCF uses nested XML structures, so the tree structure of the robot is more visible, which is important when you have many links that branch off like robot hands.
 - There are many differences between URDF and MJCF, but a feature that could help for soft robots is that you can have elasticity in the joints with the `springdamper` element
<https://mujoco.readthedocs.io/en/latest/XMLreference.html#body-joint>

Feedback to the Tutorial on Simulation – Part 2



- **Minute 16:46:** *What is the recommended directory structure for creating a project using Mujoco and the other software packages that we might use? I'd like to make sure that developing, testing, and sharing the software packages that I develop would be streamlined as possible.*
 - MuJoCo's source directory structure is described here:
<https://mujoco.readthedocs.io/en/latest/programming/>
bin - dynamic libraries, executables, MUJOCO_LOG.TXT
doc - README.txt and REFERENCE.txt
include - header files needed to develop with MuJoCo
model - model collection
sample - code samples and makefile need to build them
 - Do not worry about a project directory structure that is fit for your project, including RL etc.. Most of the code will already be ready with a provided structure for you in the RL pipeline, the code you write for the IsaacGym integration will be based on: https://github.com/srl-ethz/faive_gym_oss
 - You can refer to that for the project structure. The only thing you should be worried about right now is putting together a nice folder for your team's model - for which you can find examples here: https://github.com/google-deepmind/mujoco_menagerie.
If you want to do the structure super clean, we would advise to create a separate git repo for your robot models, then import the models into a subdirectory as a submodule.

Feedback to the Tutorial on Simulation – Part 3



- **Minute 28:07:** *What are the limitations of the SOFA simulator? They were not mentioned here in the video...*
 - We did not dive deeper into SOFA, since that software is specifically only for large deformations as found in very soft robots.
SOFA is not parallelizable (only CPU support), it runs rather slow unless the resolution of the mesh is lowered heavily, and the software requires quite some time to learn and use. For highly articulated problems with smaller amount of soft deformations, Mujoco is more suitable. Parallel Simulation can be done in Isaac Gym / Isaac Orbit.

Tutorial needs updating

- **Minute 28:44:** *During the reinforcement learning training, when there are 1000s of agents performing the task, is it possible for them to learn from each other? The best analogy I can give is similar to how evolutionary algorithms work, where whatever is optimizing to the goal is kept, mutated a bit, and passed to the other agents. Also, what would be the downside of such approach?*
 - The 1000s of agents all work with the same RL policy and so the learnings from all simulations flow into the weight updates of the same RL policy.
 - The sole reason we are using this amount of parallel environments is that we can collect more interactions with the simulator. A more detailed discussion can be found here:
<https://ai.stackexchange.com/questions/12059/how-is-parallelism-implemented-in-rl-algorithms-like-ppo>

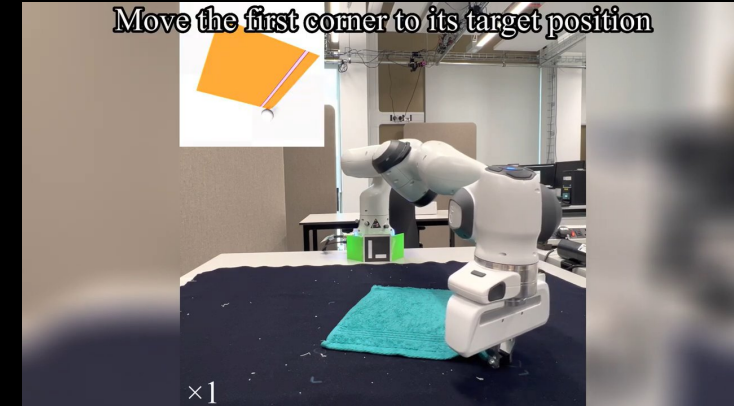
Some example object manipulation applications



Task-based grasping



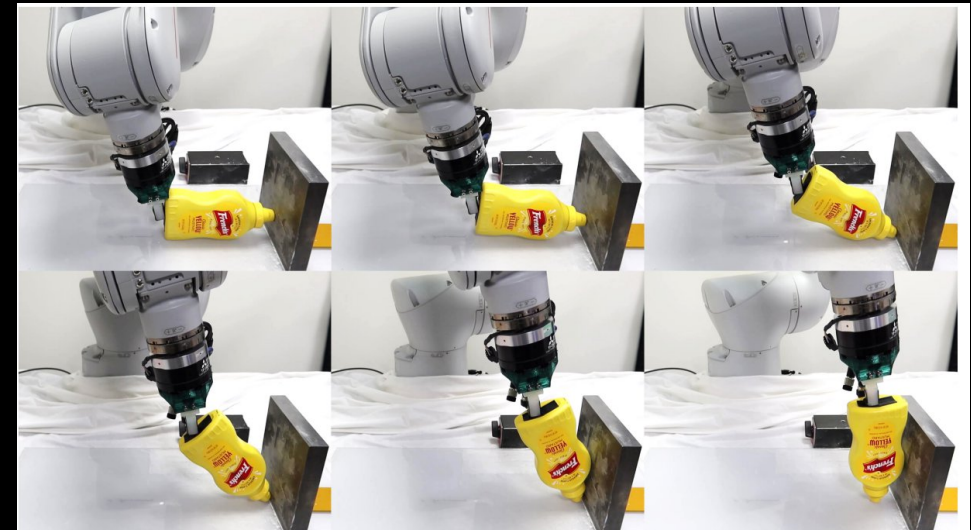
Deformable object manipulation



Manipulation in clutter



Manipulation with environmental contact



Zhang et al. ICRA 2023

Change: Workshop on Interfacing Motors and Your Hand will happen next week!



- Today: you receive the motors
 - Next week: you learn how to interface the motors
 - Wait to use them
 - Bring the motors and (ideally) a finger to the workshop

Unit	Week	Tutorial	Class Day	Focus and Q&A Talk	Workshop
				Needs updating	
				Tuesdays, 14:15 - 15:00, CLA E 32.2	Mondays, 15:15 - 16:00, CLA E 32.2
1	2		25.09.	The Challenge - Robotics Hands for Dexterous Manipulation <i>Robert Katzschmann</i>	Showcase & Design Discussion <i>Benedek Forrai</i>
2	3	Building and Fabricating an Articulated Robot Hand	02.10.	Design of Robotic Hands <i>Robert Katzschmann</i>	Materials and Motors <i>Manuel Knecht</i>
3	4		09.10.	Prototyping and Fabrication Techniques for Robots <i>Robert Katzschmann</i>	3D-Printing and Silicon Casting <i>Stephan Gravert</i>
4	5	Simulating Robots and Soft Interaction with the World	16.10.	Methods and Challenges in Simulation <i>Robert Katzschmann</i>	Simulating with MuJoCo <i>Yasunori Tshimitsu</i>
5	6		23.10.	Hardware Check-Up and Q&A <i>Robert Katzschmann & Ph.D.s</i>	Interfacing & Controlling your Robotic Hand <i>Manuel Knecht</i>
6	7	Identifying Kinematics and Dynamics of Robots	30.10.	Implementation Dynamics of Robotic Hands <i>Robert Katzschmann</i>	Hardware Check-Up and Q&A <i>Robert Katzschmann & Ph.D.s</i>
7	8	Implementing Control Strategies for Manipulation	06.11.	Control Challenges for Dexterous Manipulation <i>Robert Katzschmann</i>	Interfacing & Controlling a Robotic Arm <i>Gavin Cangan</i>
8	9	Teleoperation using Machine Learning and Computer Vision	13.11.	The Benefit of Machine Learning for Robot Control <i>Robert Katzschmann</i>	Teleoperating for Manipulation <i>Elvis Nava</i>
9	10	Advancing Robust Controllers with Reinforcement Learning	20.11.	Next Steps in Machine Learning for Robotics <i>Robert Katzschmann</i>	Reinforcement Learning with Isaac Gym <i>Yasunori Tshimitsu</i>
10	11		27.11.	Software Check-Up and Q&A <i>Robert Katzschmann & Ph.D.s</i>	Control & RL Q&A <i>Yasunori Tshimitsu & Benedek Forrai</i>
11	12	Bringing Robots to the World	04.12.	Product Development Challenges <i>Mirko Meboldt & Robert Katzschmann</i>	Project Check-Up and Q&A <i>Robert Katzschmann & Ph.D.s</i>
12	13		11.12.	The Challenge I Presentation and Hardware Showcase	
13	14		18.12.	The Challenge II Teleoperation and Autonomy Showcase	

Details on Grading and Challenge



- Teleoperation & Manipulation
 - Steer hand with camera; robotic arm with joystick
 - 5 different tasks requiring dexterous manipulation (e.g., pick and place, world interaction, reorientation, different objects) -> surprise!
 - Graded based on success and speed
- Sphere Rotation with Reinforcement Learning
 - Train in simulation; execute open-loop on hardware
 - Simulation and execution will be graded
- Hardware presentation: the mechanical design, its cleverness, originality, robustness, etc. will be graded

#	Task	Type	When	Relative Weight		Absolute Weight
1	Challenge	Group	Unit 13	40.0%		40.0%
1.1	Teleoperation & Manipulation 5 tasks requiring dexterous manipulation				60.0%	24.0%
1.2	Sphere Rotation with Reinforcement learning Simulation and open-loop execution				40.0%	16.0%
1.3	Task X				50.0%	20.0%
2	Presentation	Group	Unit 12	20.0%		20.0%
2.1	Oral presentation				50.0%	10.0%
2.2	Hardware presentation				50.0%	10.0%
3	Report	Group	After Unit 13	10.0%		10.0%
4	Moodle Quizzes	Individual	Every Unit	20.0%		20.0%
5	Attendance	Individual	Every Unit	10.0%		10.0%

Keys are ready to be picked-up



- You received an email -> go to ServiceDesk in LEE
- Remember:
 - Do not lose the key
 - Always close the classroom
 - Only use your space
 - Use your lockers
 - Put general materials and tools back
 - Do not use materials, tools, and the robot before training
- Enjoy a coffee!
 - Bring your own cups
 - Clean the machine

Needs updating