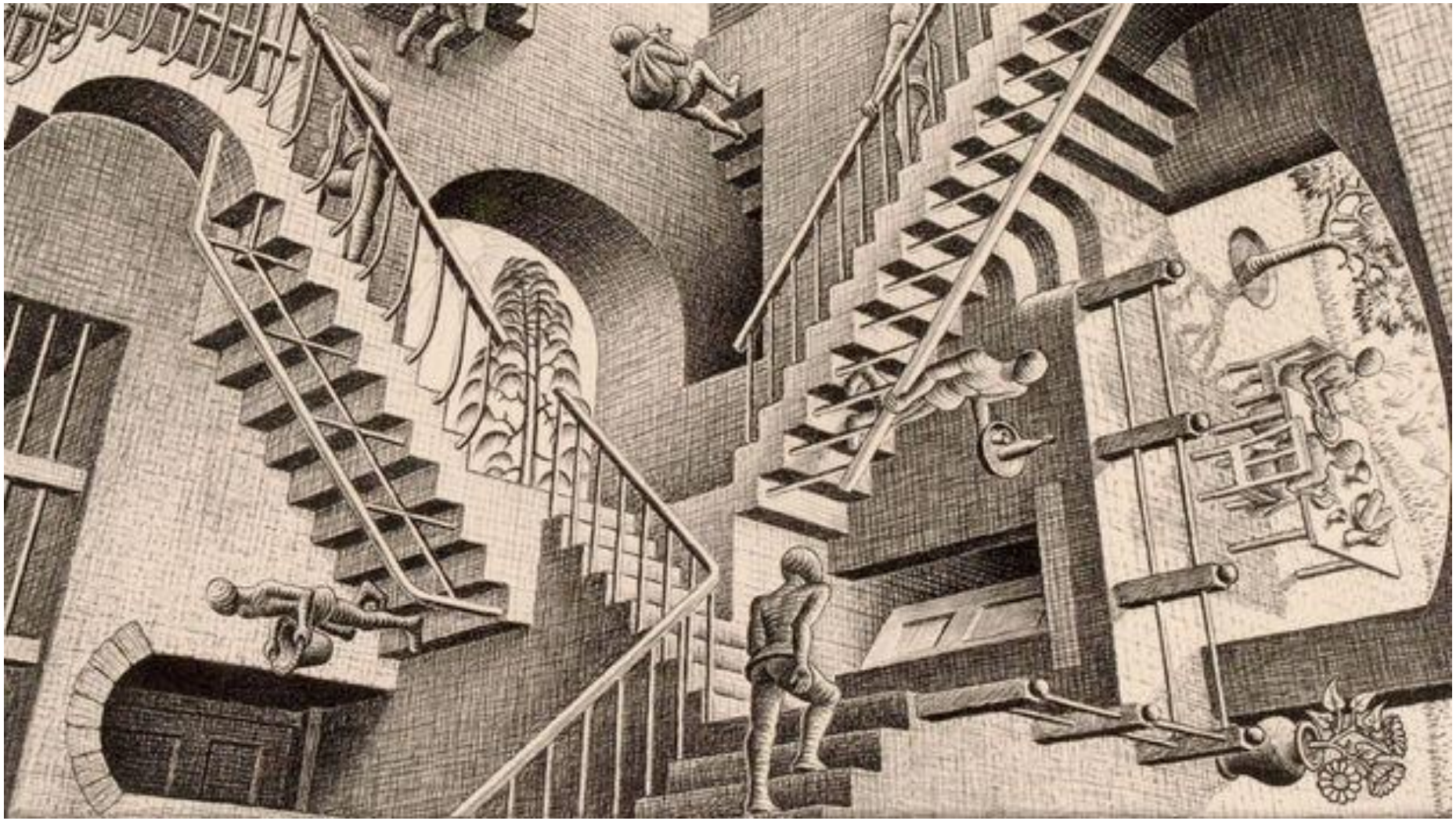
A Gentle Intro to Sim-to-Real

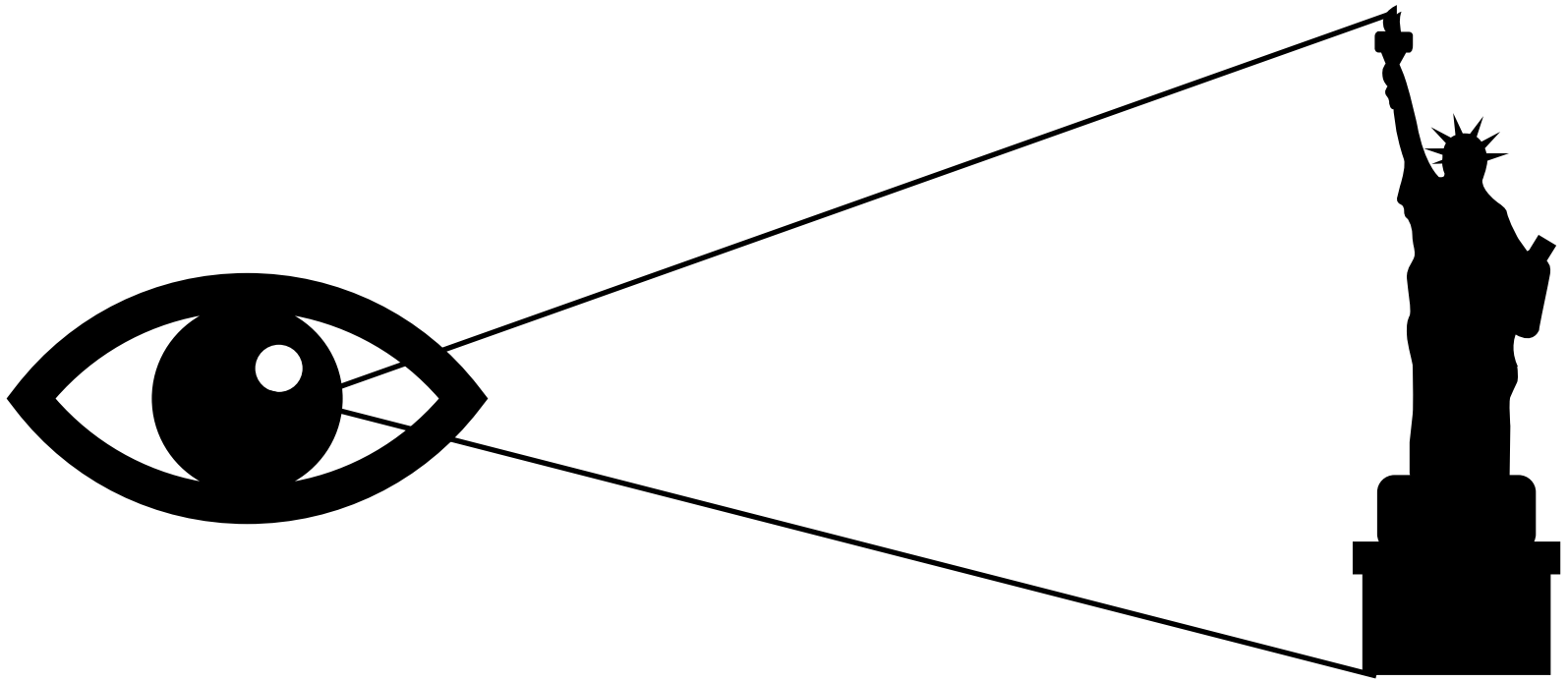
ESE 6510
Antonio Loquercio



M.C. Escher,

***Relativity*, 1953**

What is Perception?

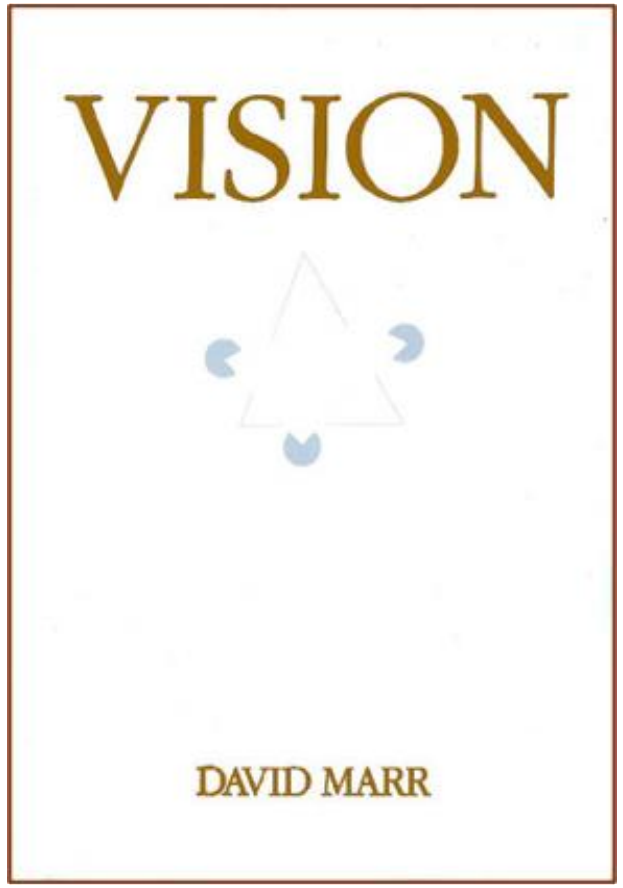


Aristotele (*De Anima*): “Perception is a kind of reception of the form of the perceived object without the matter.”

Vision (1982) By David Marr

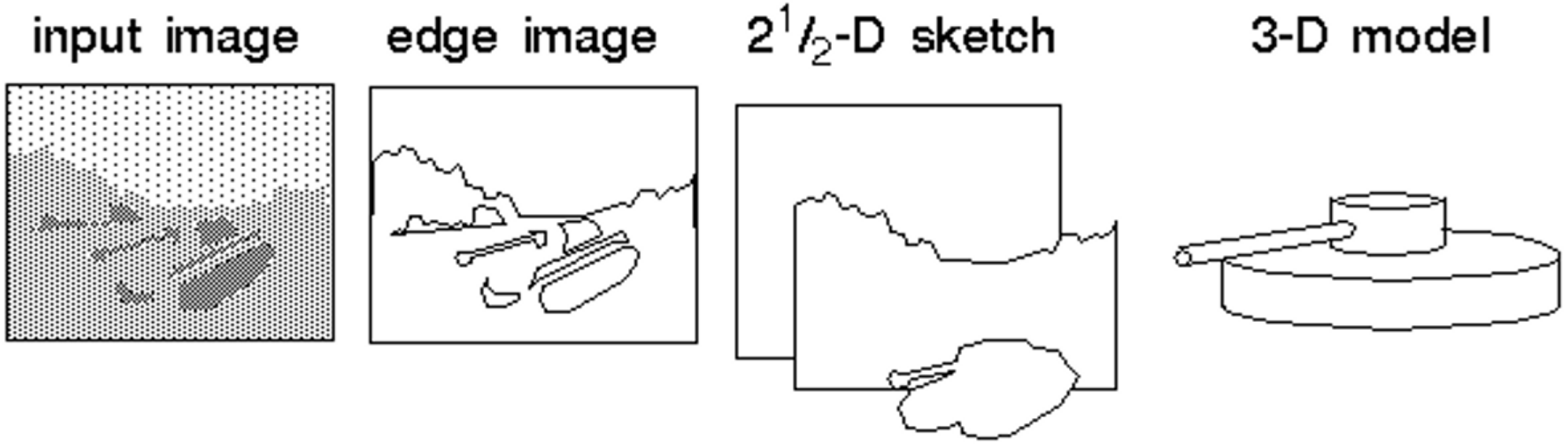
- PhD in Neuroscience, Cambridge
- Professor of Psychology at MIT (1977-1980)
- Posthumous book: Vision (1982)

In December 1977, certain events occurred that forced me to write this book a few years earlier than I had planned. Although the book has important gaps, which I hope will soon be filled, a new framework for studying vision is already clear and supported by enough solid results to be worth setting down as a coherent whole.



An *Information Processing* Theory of Vision

Vision as Inverse Graphics or Inverse Optics

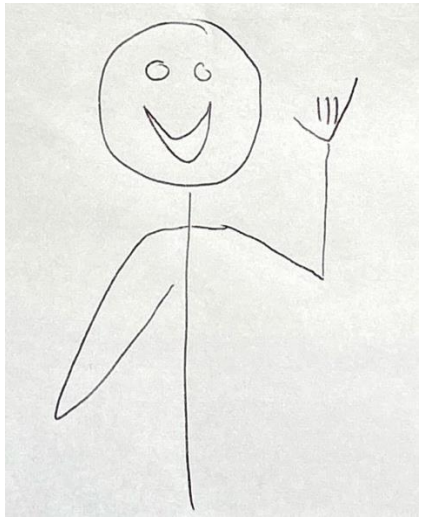


- Very appealing from a “software engineering” perspective.
  - Modularity, feedforward pipeline

The Theory Does not Fit The Data

- Humans do not recover veridical, task-independent 3D representations
- Principles of modularity and feedforward processing don’t hold for human vision

What we *think* we see



What we *really* see

W. Warren, Does This Computational Theory Solve the Right Problem? Marr, Gibson, and the Goal of Vision, Perception 41(9), 2012

Our Brain 200 Million Years Ago

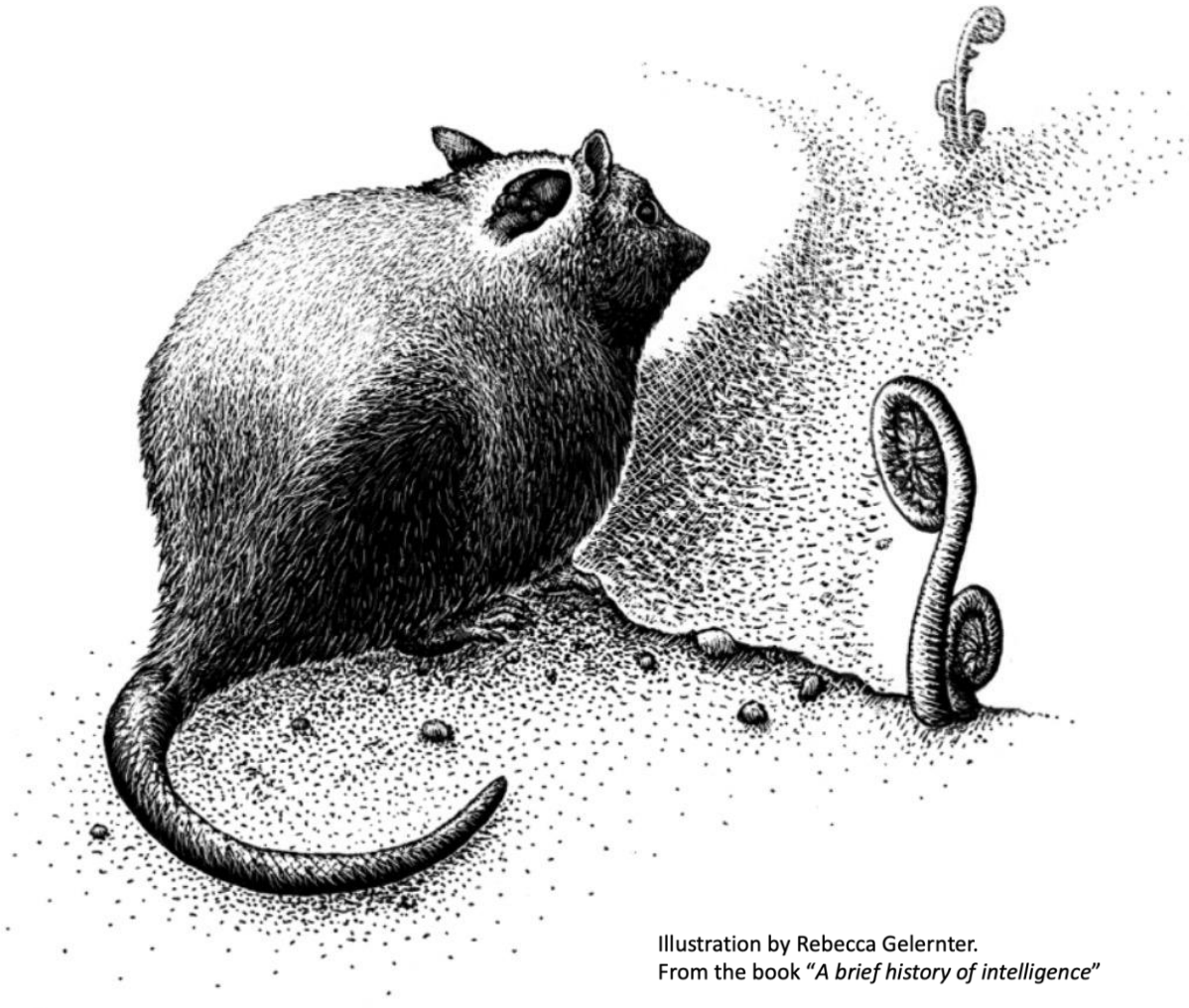
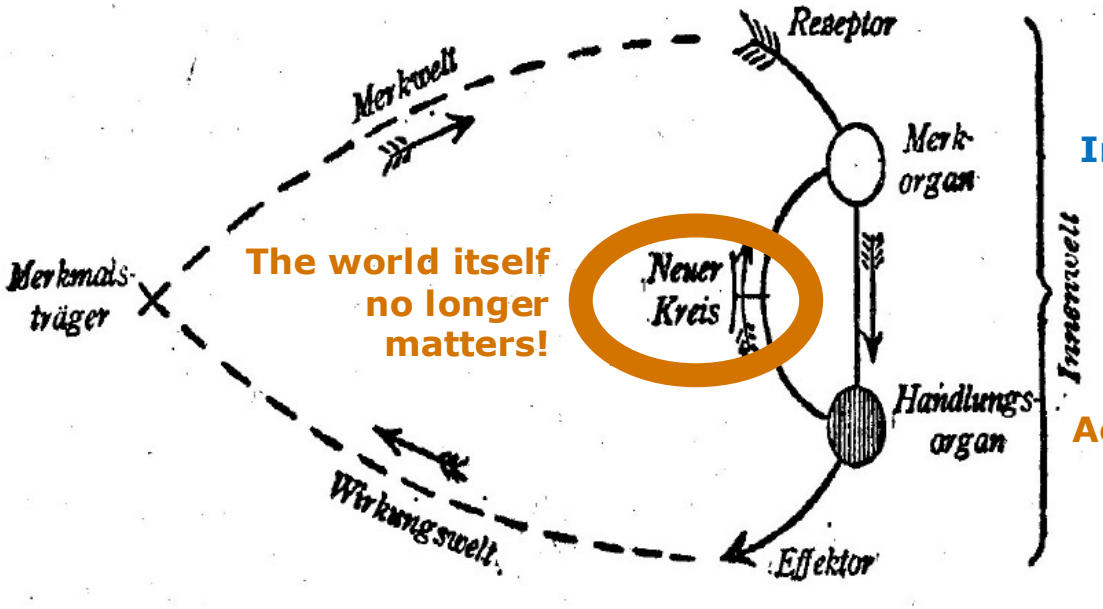


Illustration by Rebecca Gelernter.
From the book "*A brief history of intelligence*"

- Simulate the outcome of actions before they occur.
- Enabled by a neurological breakthrough: the neocortex.
- 76% of our brain consists of the neocortex.

An Alternative Theory of Perception

- Jakob von Uexküll, German biologist (1864-1944)
- Theory of “Umwelt” (Sensory-action surrounding worlds).



Feedback about the action

The world itself no longer matters!

Input from the world

Action affects the world

Differences between organisms

Reflex agent

- Consider how the world IS
- Choose action based only on current percept
- Do not consider the future consequences of actions

Predictive agent

- Consider how the world WOULD BE
- Decisions based on (hypothesized) consequences of actions
- Must have a model of how the world evolves in response to actions

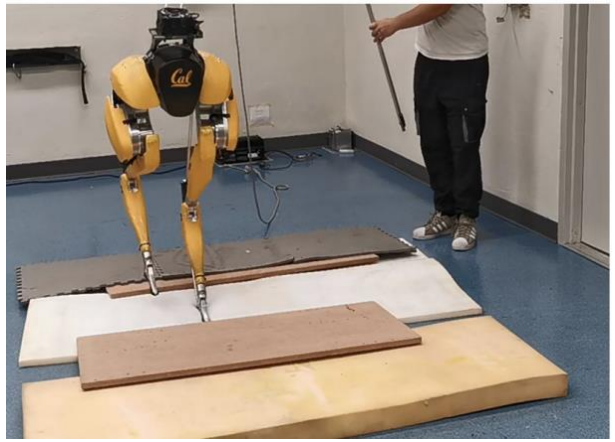
An Actionable Theory of Perception

- Natural selection optimizes fitness, not veridicality.
- The model of the world that an organism builds from perception is imperfect in many ways, but useful.
- Discrepancies between the predictions and the observed state of the world generate “sparks of awareness” (a view held by Erwin Schrödinger)

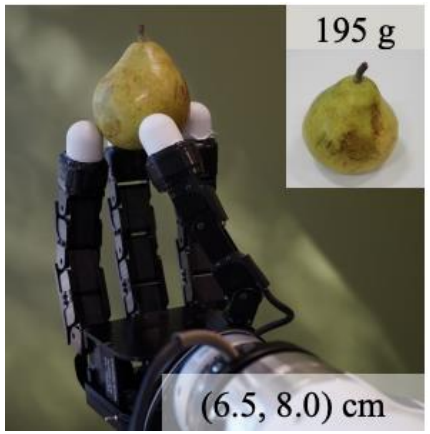
A Quiet Revolution in Robotics (2020-2022)

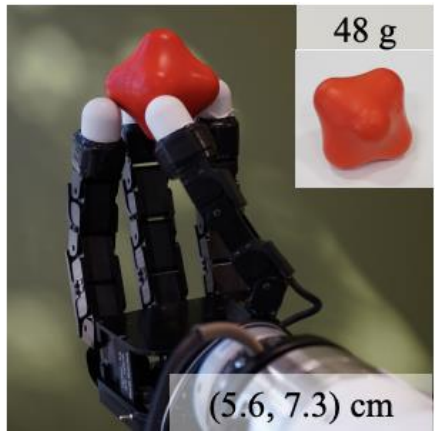
Loquercio et al, 2021

Lee et al, 2020

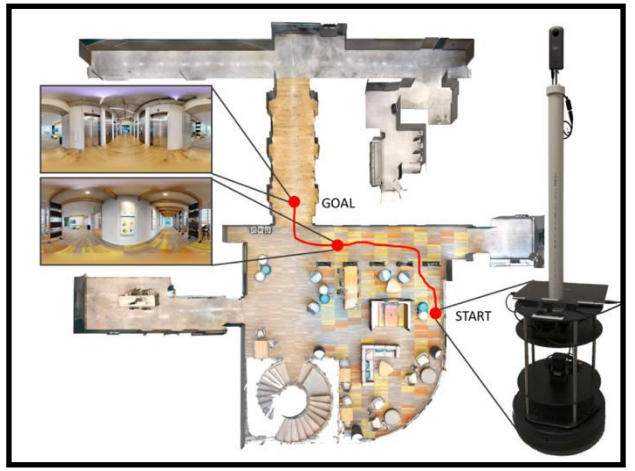


Kumar et al, 2022





Qi et al, 2022



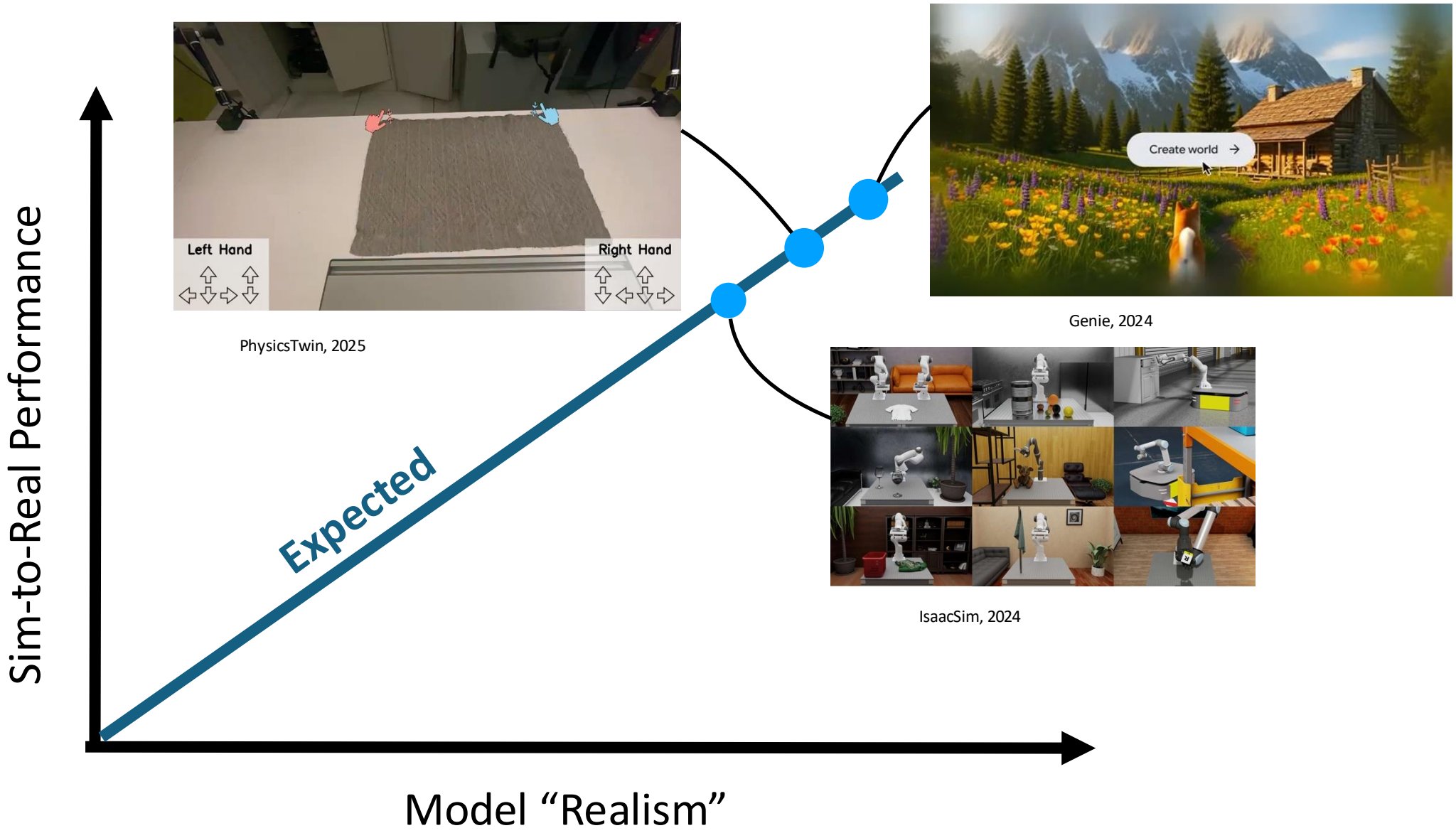
Anderson et al, 2021

Simulation-to-Real-World Transfer

Learning High-Speed Flight in the Wild, Science Robotics 2021

Antonio Loquercio, Elia Kaufmann, Rene Ranflt, Matthias Mueller, Vladlen Koltun, Davide Scaramuzza

Towards Hyper-Realistic Models



PhysicsTwin, 2025

Genie, 2024

IsaacSim, 2024

A Powerful Argument Against Simulators

The computational power of the physical world scales linearly with the number of particles. Think of pouring sand. Every sand particle computes its interactions with other particles. The more particles, the more computation. Thus the amount of computation scales linearly with the number of particles. **None of our computer devices can provide such computational power.** Assuming that accurate simulation oh physical behavior is important for robotics, this appears to be a fundamental challenge for the simulation approach.

By Rodney Brooks

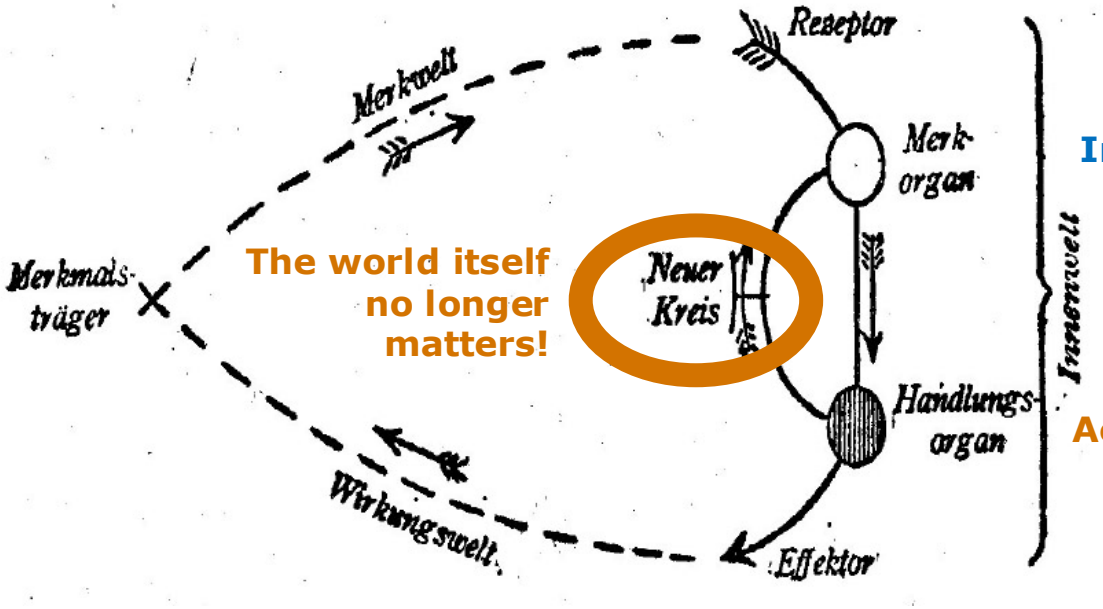
A Powerful Argument Against Simulators

The computational power of the physical world scales linearly with the number of particles. Think of pouring sand. Every sand particle computes its interactions with other particles. The more particles, the more computation. Thus the amount of computation scales linearly with the number of particles. **None of our computer devices can provide such computational power. Assuming that accurate simulation oh physical behavior is important for robotics**, this appears to be a fundamental challenge for the simulation approach.

By Rodney Brooks

An Alternative Theory of Perception

- Jakob von Uexküll, German biologist (1864-1944)
- Theory of “Umwelt” (Sensory-action surrounding worlds).



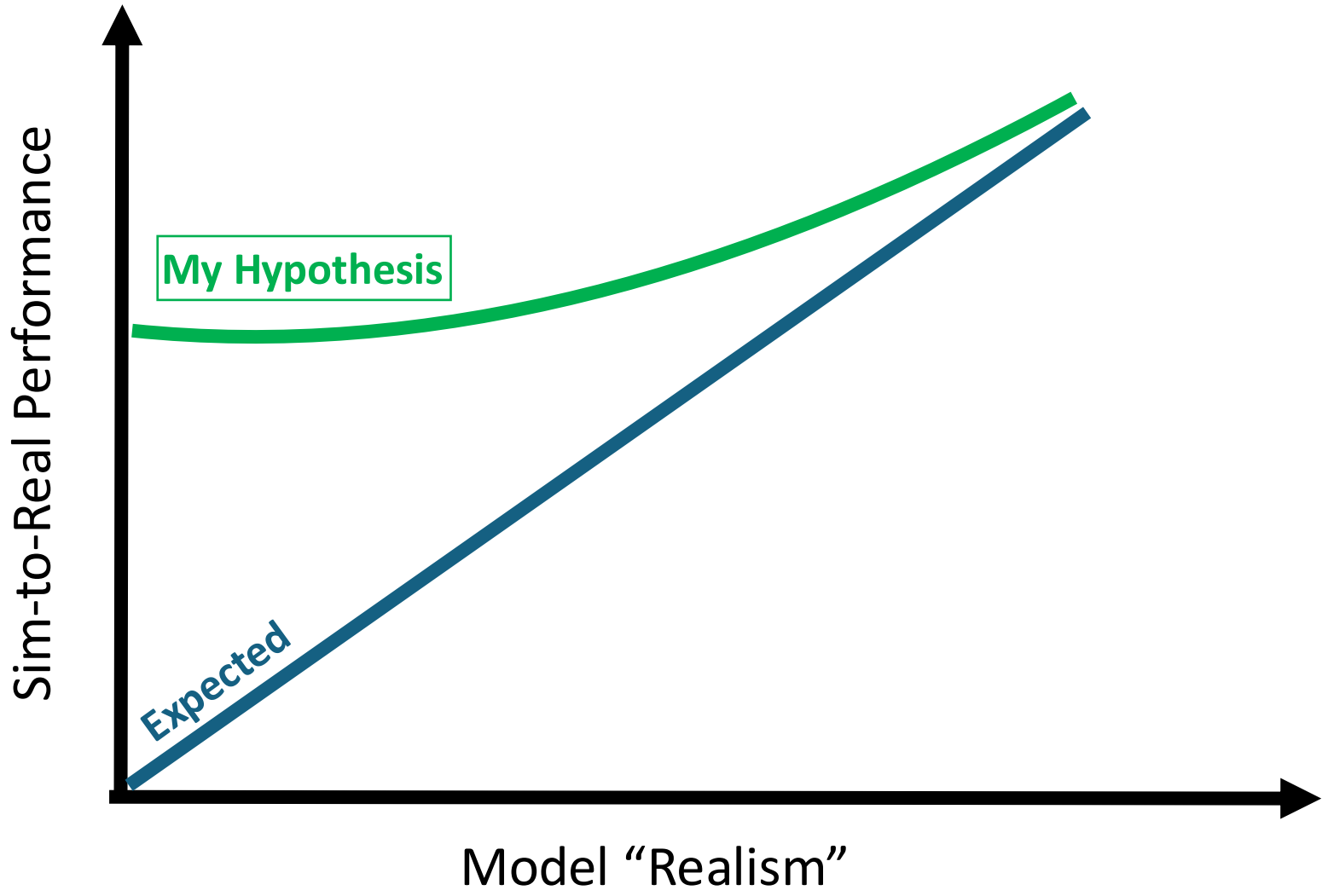
Feedback about the action

The world itself no longer matters!

Input from the world

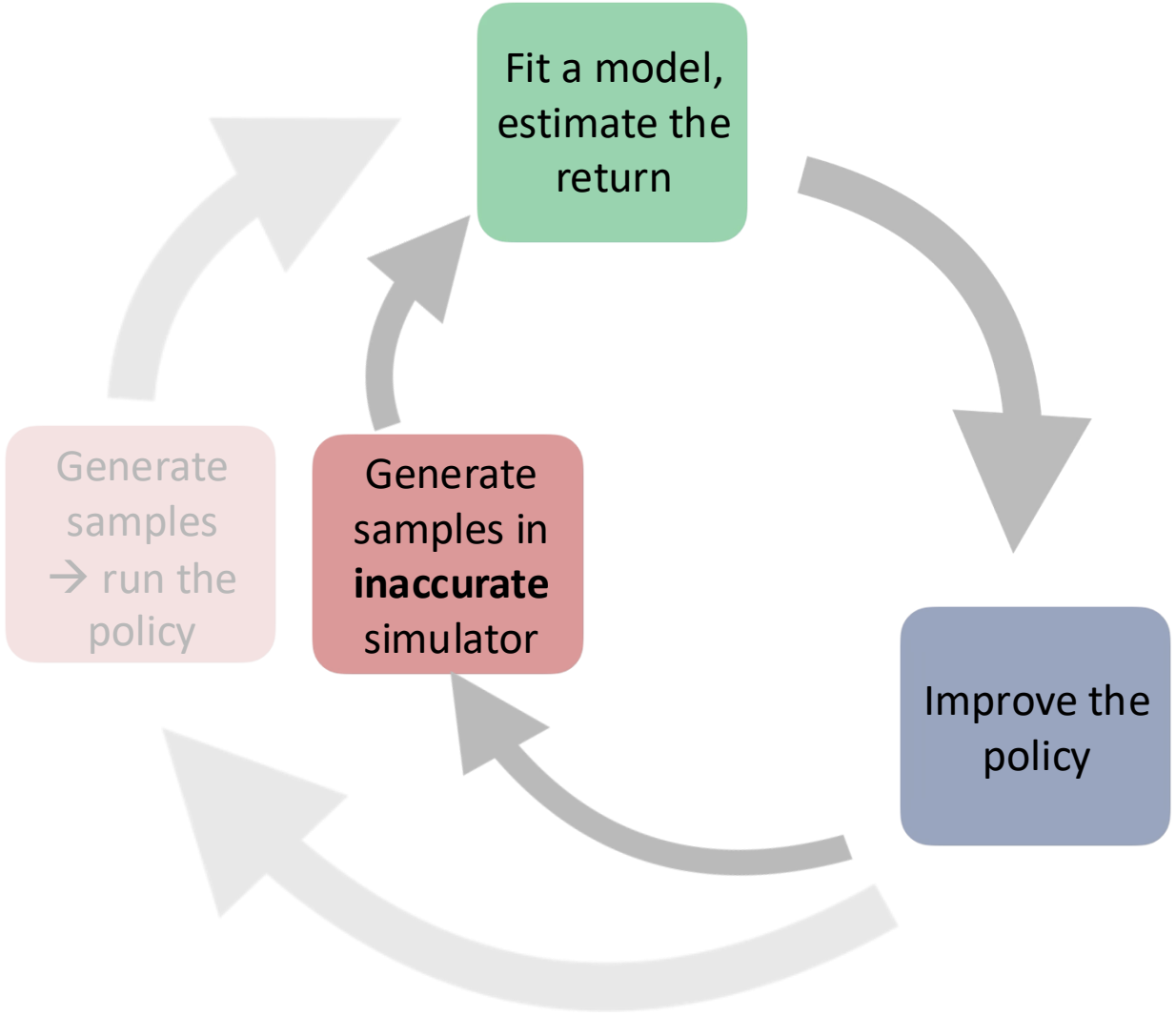
Action affects the world

Sensorimotor Reduced-Order Models



What is Sim-to-Real?

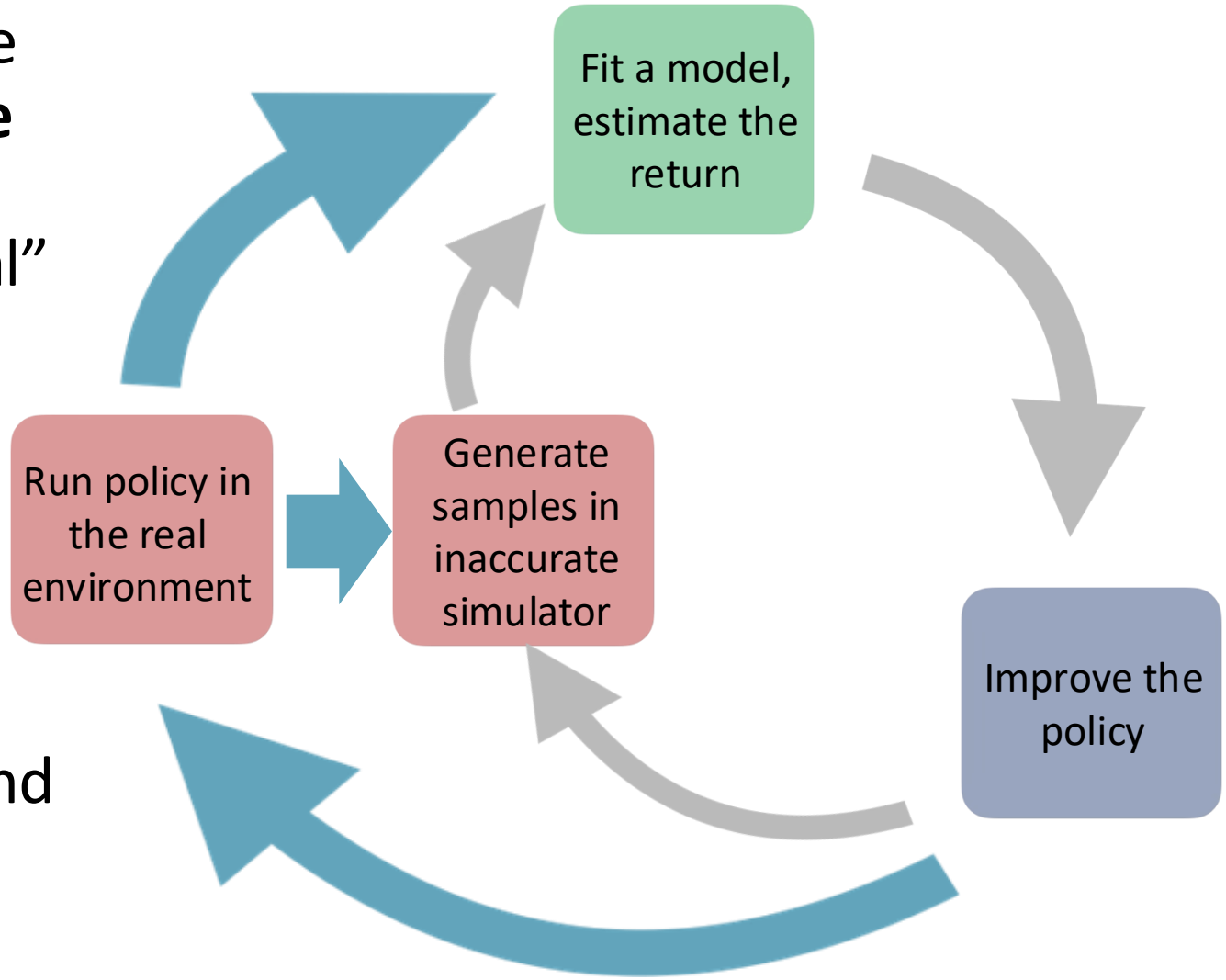
A family of methods that enable policies trained in an **inaccurate world model** to successfully transfer and operate in the “real” environment.



What is Sim2Real?

A family of methods that enable policies trained in an **inaccurate world model** to successfully transfer and operate in the “real” environment.

Evaluations of the policy in the real environment are used to **improve** both the **simulation** and the **policy learning** approach.



What Sim-to-Real is NOT

- Equivalent to model-based RL:
  - Model-based RL is a super-set of sim2real. The latter is specifically tailored to imperfect simulators.
- A way to train policies with PPO in physics-based simulators.
  - Why only PPO? Can't I train a policy with other methods (e.g., imitation)?
  - Why should the simulator be physics-based? Can be data-driven or hybrid.
- A trick to train policies in fast simulators
  - You could have advantages even if the model is slower than the real environment.

Why Sim-to-Real?

- Faster/Safer to run a policy
  - Especially important for partially trained policies
- Focus policy learning on the subset of the agent/environment dynamics that matter the most for the task at hand.
- Fine-grained control of policy experience/curriculum.
- Explore counterfactuals (what if one of the legs breaks?)

The Spectrum of Simulators (or World Models)

Physics-based

Data-Driven

GENESIS

Too many to list

MuJoCo

ManiSkill

...

General / Ease of Customization / Faster

Visual Fidelity / Environment Diversity

The Simulation to Reality Gap(s)

- **Dynamics gap**:

$$s_{t+1} = f_{real}(s_t, a_t) \neq f_{sim}\,(s_t, a_t)$$

- **Sensory gap**:

$$\mathrm{o_t} = obs_{real}(s_t) \neq obs_{sim}(s_t)$$

- **Semantic (or Environment) gap**:
  - Things should have the right size relatively to each other and placed where it makes sense.
  - Other agents should behave in a way that is realistic.

The Semantic (or Environment) Gap

The Simulation to Reality Gap(s)

- Categorizing why the sim-to-real gap arises:
  - Dynamics gap
  - Sensory gap
  - Semantic (or Environment) gap

- They are all related to each other.
  - Example: what is in the environment influences what the sensors see.

- However, the taxonomy is useful for thinking about what an approach can or cannot do.

Categories of Sim-to-Real Algorithms

- Domain Randomization
  - Naïve
  - Adaptive
  - Physics-based
- System Identification
  - Explicit System Identification
  - Implicit System Identification
- Abstractions-based
  - Visual Abstractions
  - Control Abstractions

Most papers use a mix of these techniques.

Domain Randomization

- **Key Idea**: Randomize the parameters of the simulator that are important for my task and I am uncertain of.

Dynamics

Examples: Mass, Friction, Restitution Coefficient. Motor Efficiency

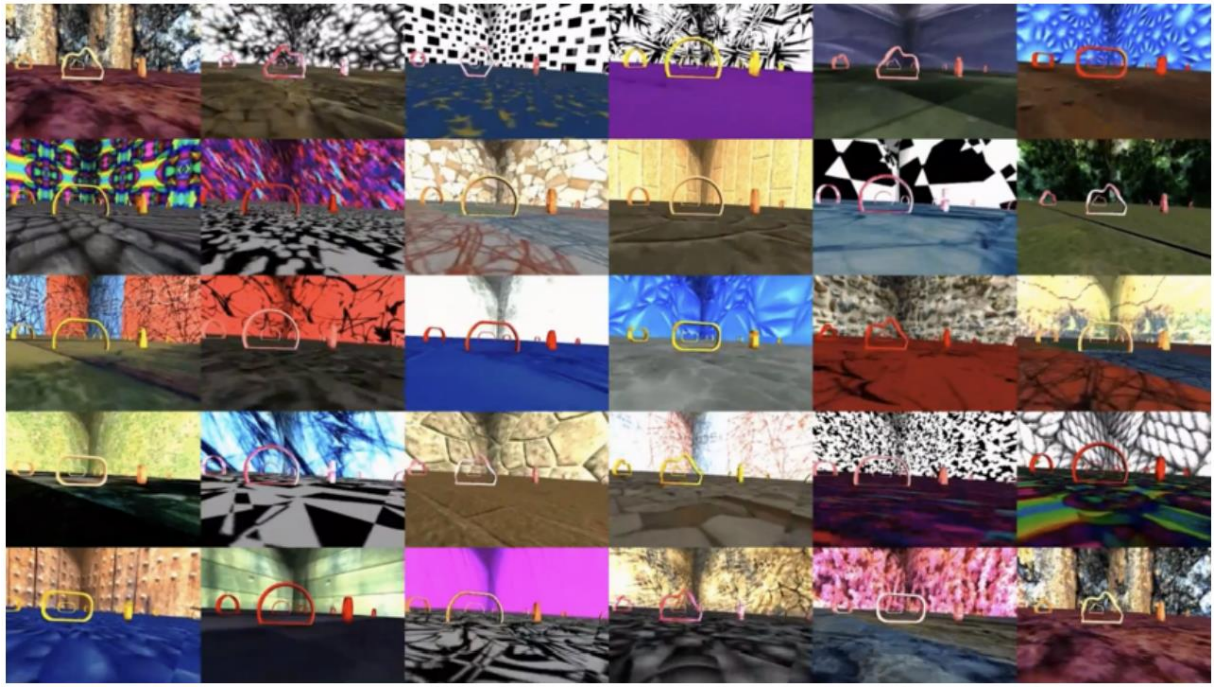
Sensory

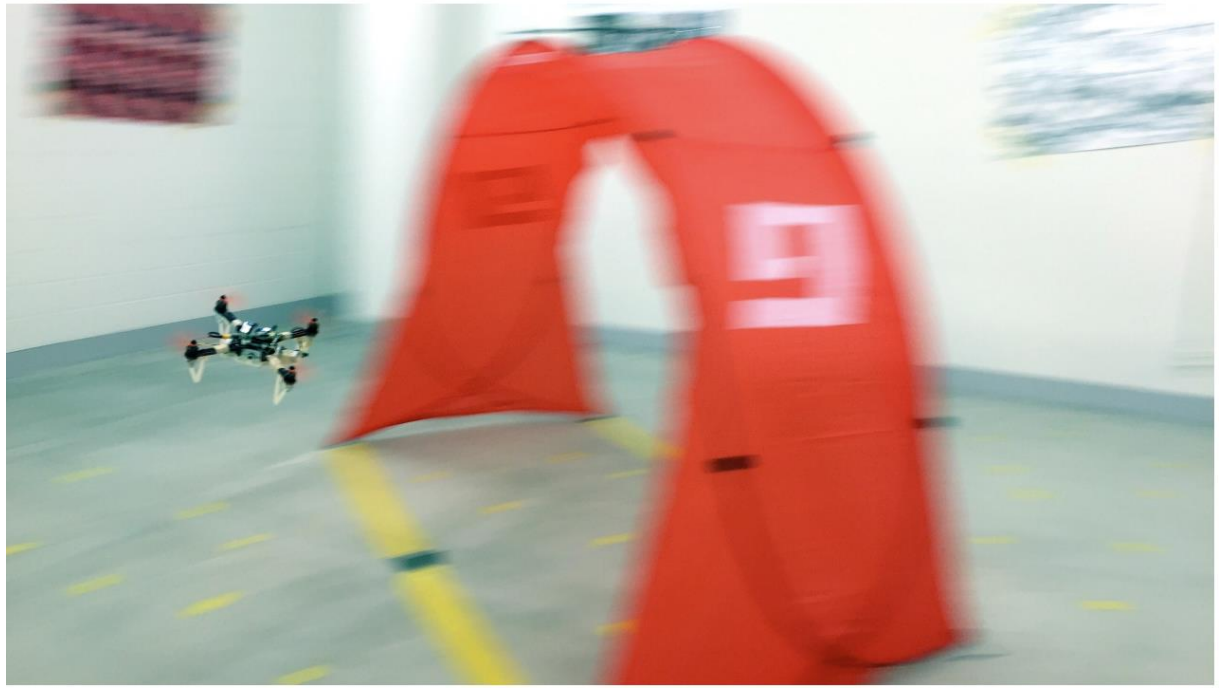
Examples: Latency, sensor noise, accuracy

Semantics

Examples: Background, Illumination, Object Types & Location

Domain Randomization: An Example





Deep Drone Racing: from Simulation to Reality with Domain Randomization, T-RO 2019

Domain Randomization

- Objective: maximize the expected RL cost given the distribution of environment parameters.

$$\theta^* = argmax_\theta \, \mathbb{E}_{e \sim D}[\, C(\theta, e)]$$

Where:

- θ are the (trainable) parameters of the policy
- e are the environment parameters (friction, mass, etc.), i.e., $s_{t+1} = f(s_t, a_t, e)$.
- D is the distribution of parameters we sample from.
- $C(\theta, e) = \mathbb{E}_{\tau(e,\theta)}[r(s_t, a_t)]$ is the RL cost.

Domain Randomization vs Robust Control

- Domain Randomization maximizes the objective in expectation:

$$\theta^* = argmax_\theta \, \mathbb{E}_{e\sim D}[\, C(\theta, e)]$$

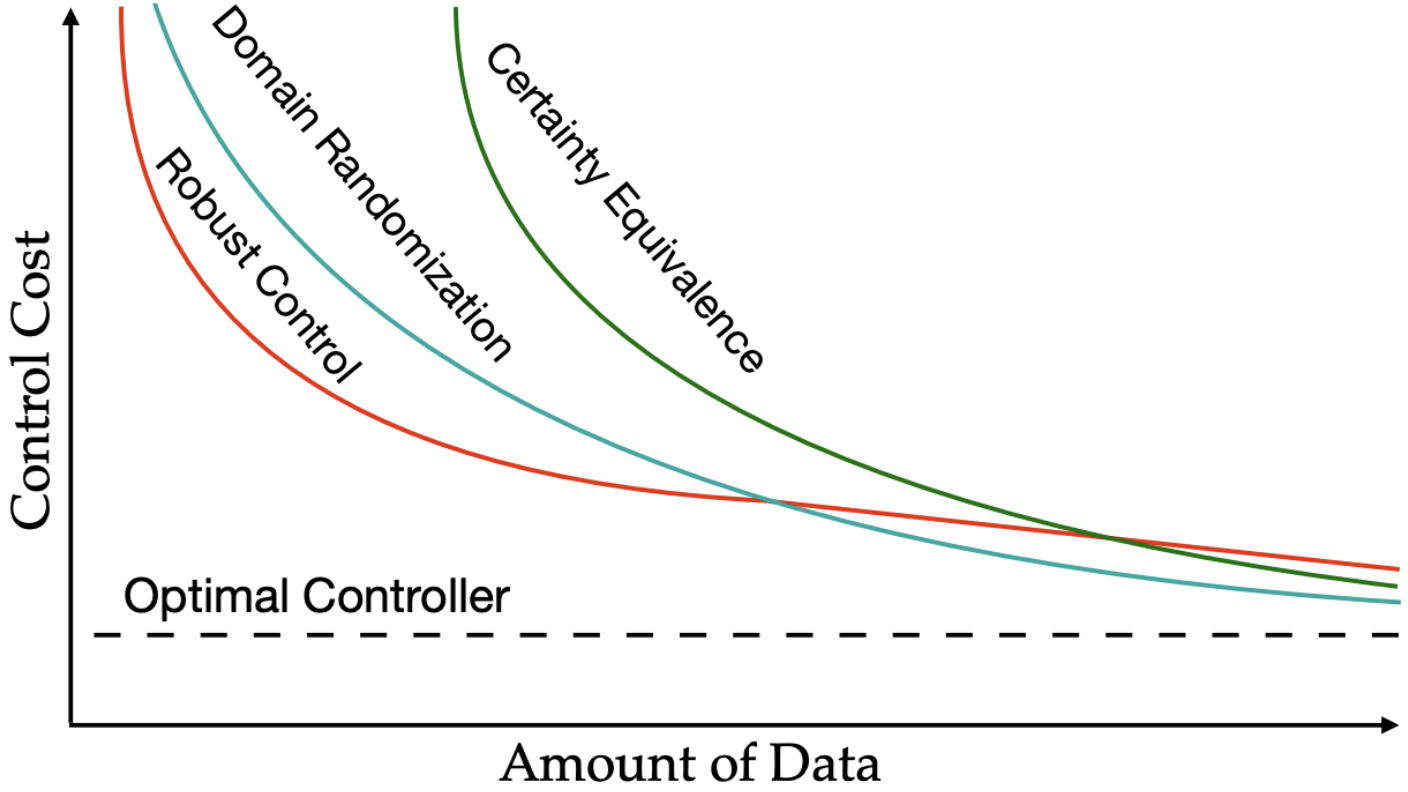
- Robust control maximizes the worst-case scenario of the objective.

$$\theta^* = argmax_\theta \, min_e [C(\theta, e)]$$

- **Domain Randomization is less conservative than Robust Control**.
- In both cases, the larger the support of the parameter distribution, the harder the task.

Burn-in time vs Cost at Convergence

You can theoretically and empirically show that the relation below holds below for a linear system and LQR controller (no RL).



Domain Randomization is Sample Efficient for Linear Quadratic Control, 2025

How to pick the parameters distribution?

- Naïve
- Performance-based (Adaptive)
- Physics-based

How to pick the parameters distribution?

- Naïve: Select the subspace of parameters you are the most uncertain about. Fix the ones you know.

- If no prior information is available, use a uniform distribution over the parameter range.

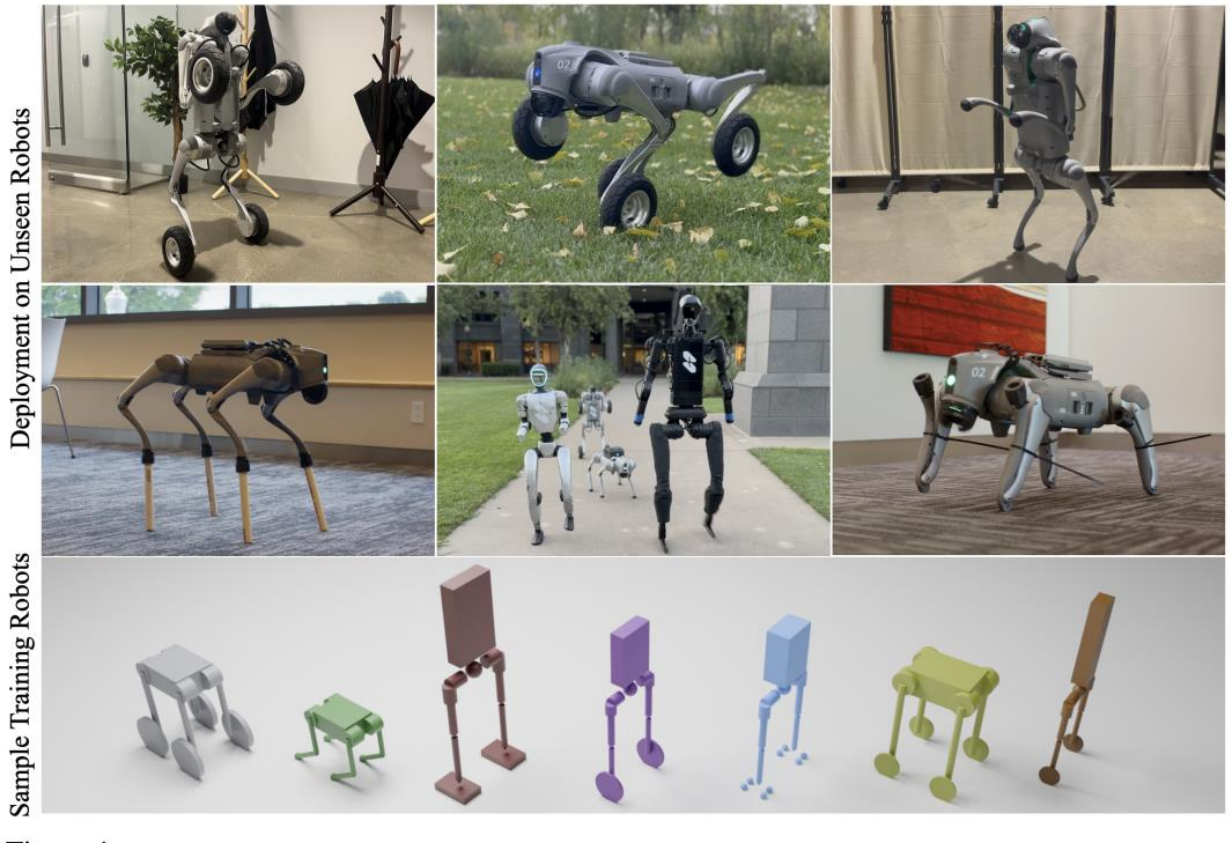
Limitations?

- Compute-intensive/challenging to converge if the ranges are large.
- Large ranges often lead to conservative policies.

Naïve DR is the Standard in Robotics

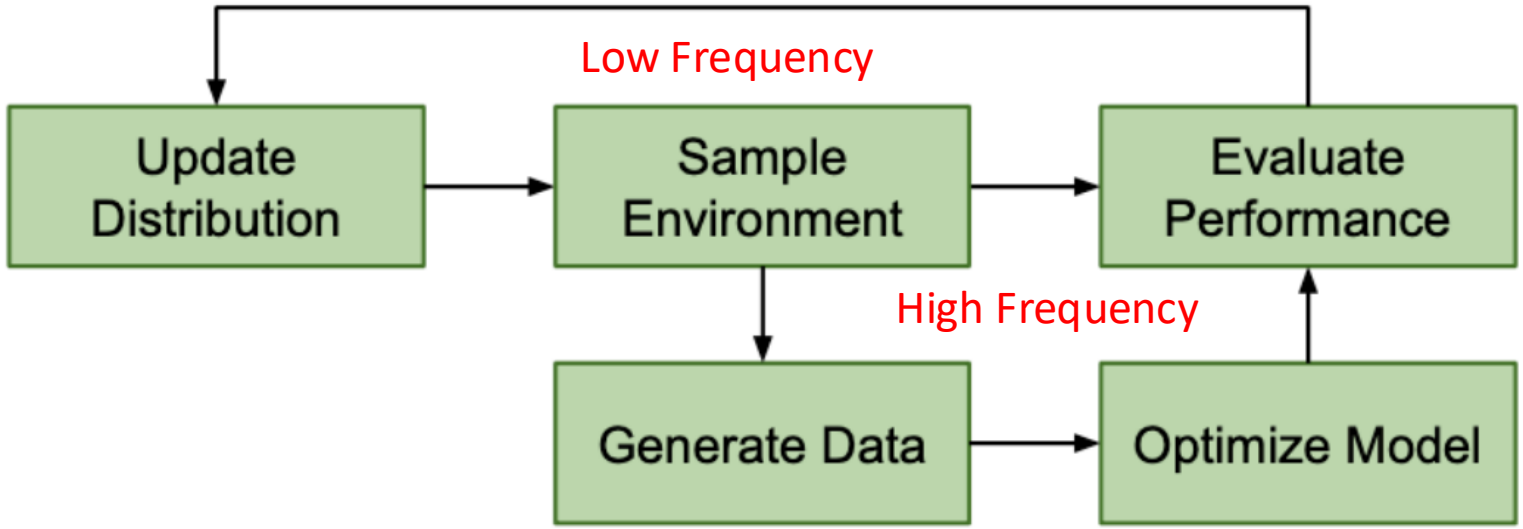
LocoFormer: Generalist Locomotion via Long-context Adaptation

Min Liu **Deepak Pathak** **Ananye Agarwal**
Skild AI



How to pick the parameters distribution?

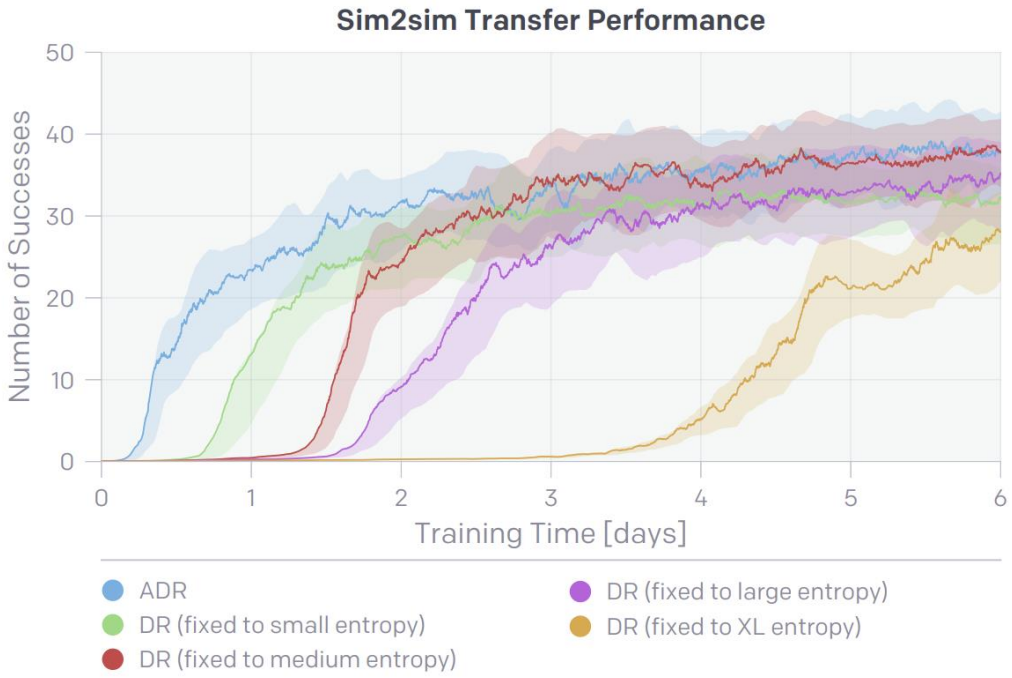
- Performance-based (adaptive):
  - Gradually adjust the ranges so the agent is never too successful or too unsuccessful.
  - Automatic Curriculum.

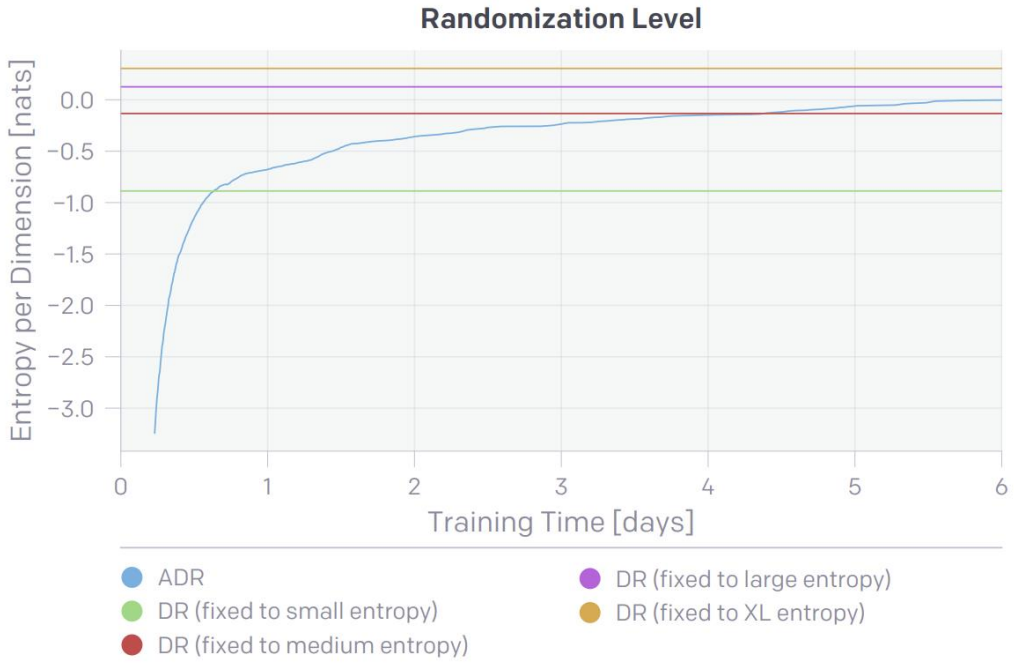


Solving Rubik's Cube with a Robot Hand, 2019

How to pick the parameters distribution?

- Performance-based (adaptive): Faster convergence than naïve DR





Solving Rubik's Cube with a Robot Hand, 2019

How to pick the parameters distribution?

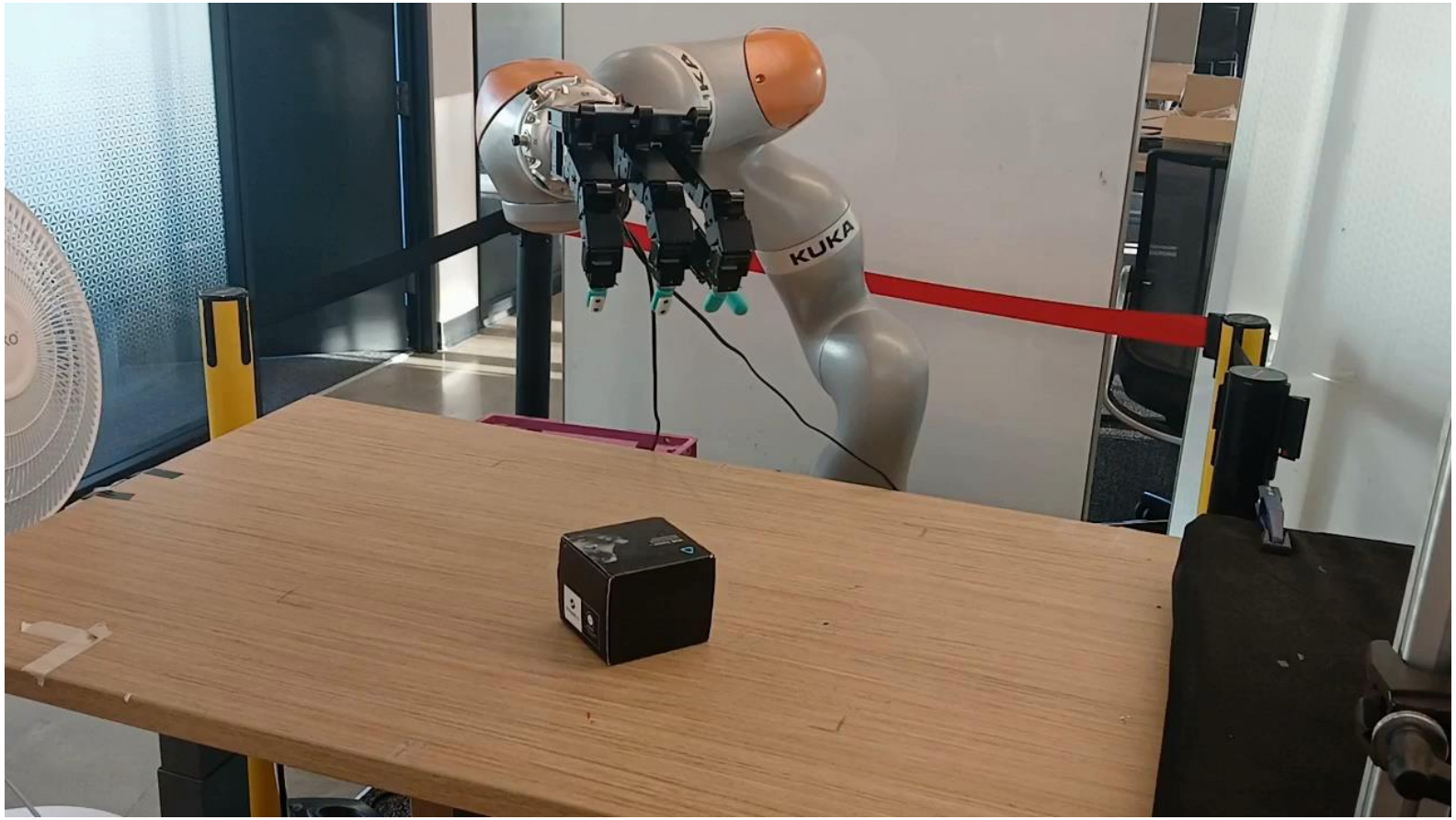
- Performance-based (adaptive): Can potentially train on larger ranges and therefore transfer better.

Policy	Training Time	ADR Entropy	Successes (Sim)		Successes (Real)	
			Mean	Median	Mean	Median
Baseline (data from [77])	—	—	43.4 ± 0.6	**50**	18.8 ± 5.4	13.0
Baseline (re-run of [77])	—	—	33.8 ± 0.9	**50**	4.0 ± 1.7	2.0
Manual DR	13.78 days	-0.348^{*} npd	42.5 ± 0.7	**50**	2.7 ± 1.1	1.0
ADR (Small)	0.64 days	-0.881 npd	21.0 ± 0.8	15	1.4 ± 0.9	0.5
ADR (Medium)	4.37 days	-0.135 npd	34.4 ± 0.9	**50**	3.2 ± 1.2	2.0
ADR (Large)	13.76 days	0.126 npd	40.5 ± 0.7	**50**	13.3 ± 3.6	11.5
ADR (XL)	—	0.305 npd	45.0 ± 0.6	**50**	16.0 ± 4.0	12.5
ADR (XXL)	—	**0.393 npd**	$\mathbf{46.7 \pm 0.5}$	**50**	$\mathbf{32.0 \pm 6.4}$	**42.0**

Solving Rubik's Cube with a Robot Hand, 2019

How to pick the parameters distribution?

- Performance-based (adaptive): Not very much used in practice (but I have seen some good recent papers using it).



End-to-end RL Improves Dexterous Grasping Policies, 2025

How to pick the parameters distribution?

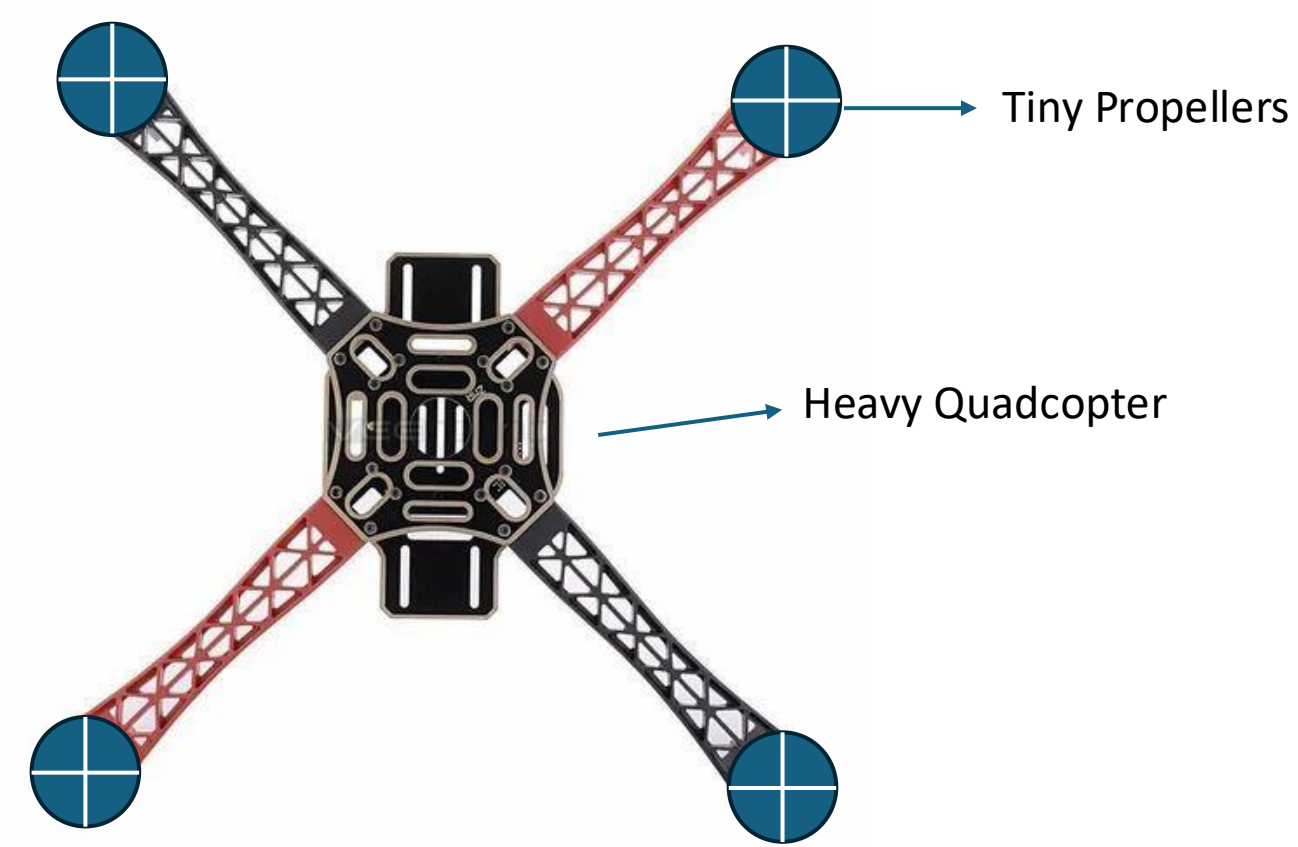
- Performance-based (adaptive): Not very much used in practice (but I have seen some good recent papers using it).

- Limitations?

- Many more hyperparameters beyond the parameter ranges
  - Success/Failure thresholds, minimum waiting time, delta update, …
- To some extent, this happens already if you train with massively parallel environments:
  - At the beginning of training, the batch will be full of “easy” parameters, but later it will balance.

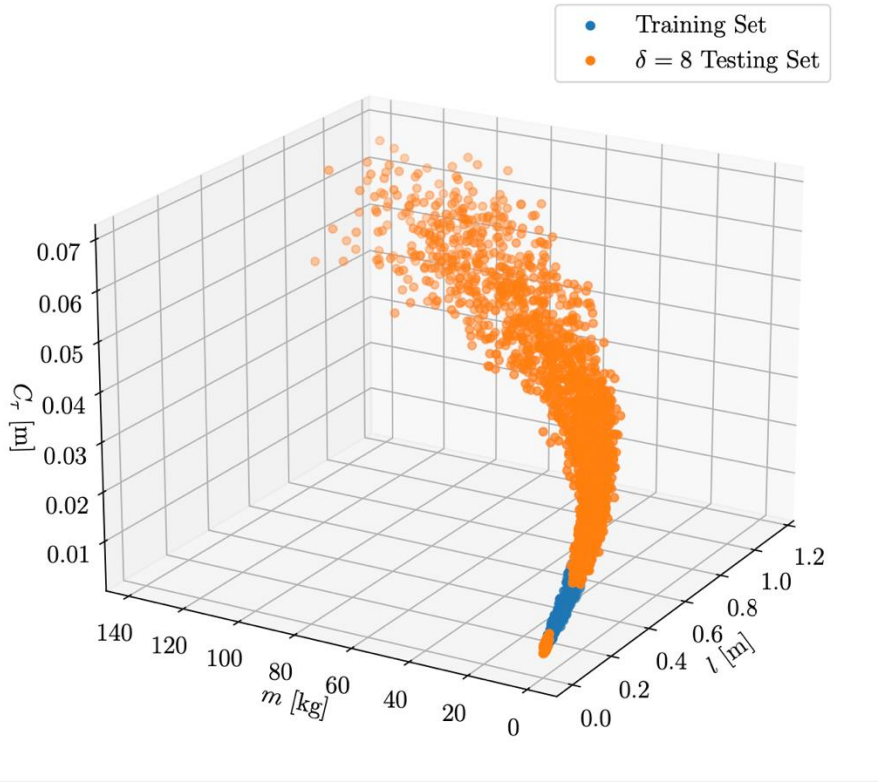
How to pick the parameters distribution?

- Physics-based: Use prior knowledge about the system to generate parameter combinations that are physically plausible.
- Why train on this?



How to pick the parameters distribution?

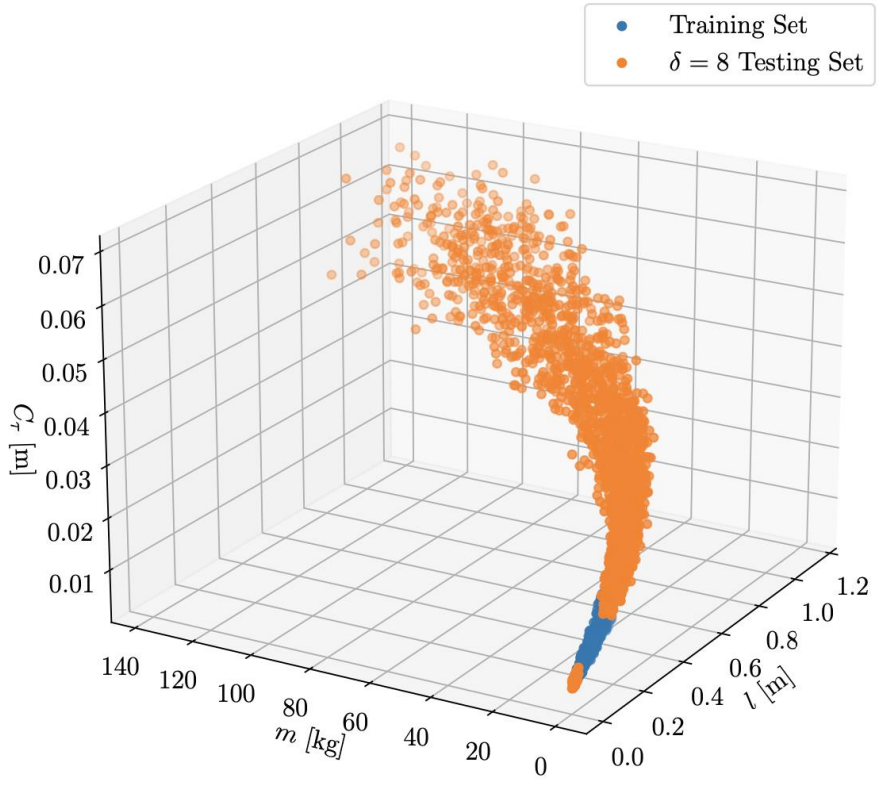
- Physics-based: Use prior knowledge about the system to generate parameter combinations that are physically plausible.

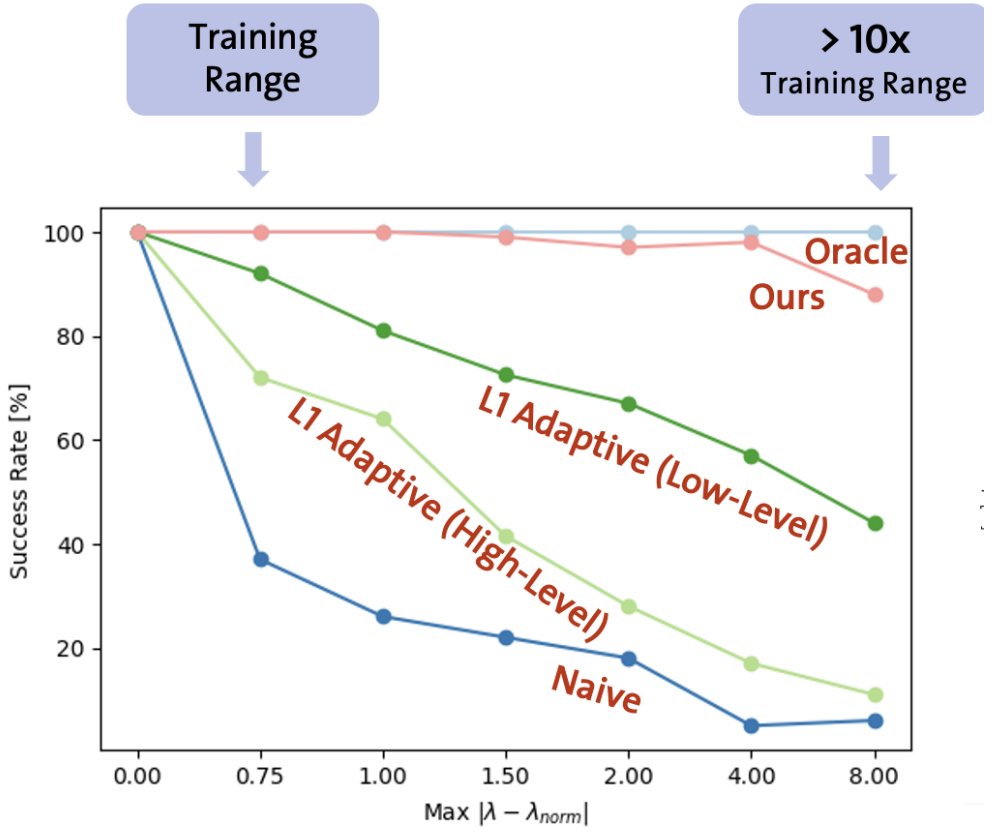


A Learning-based Quadcopter Controller with Extreme Adaptation, 2025

How to pick the parameters distribution?

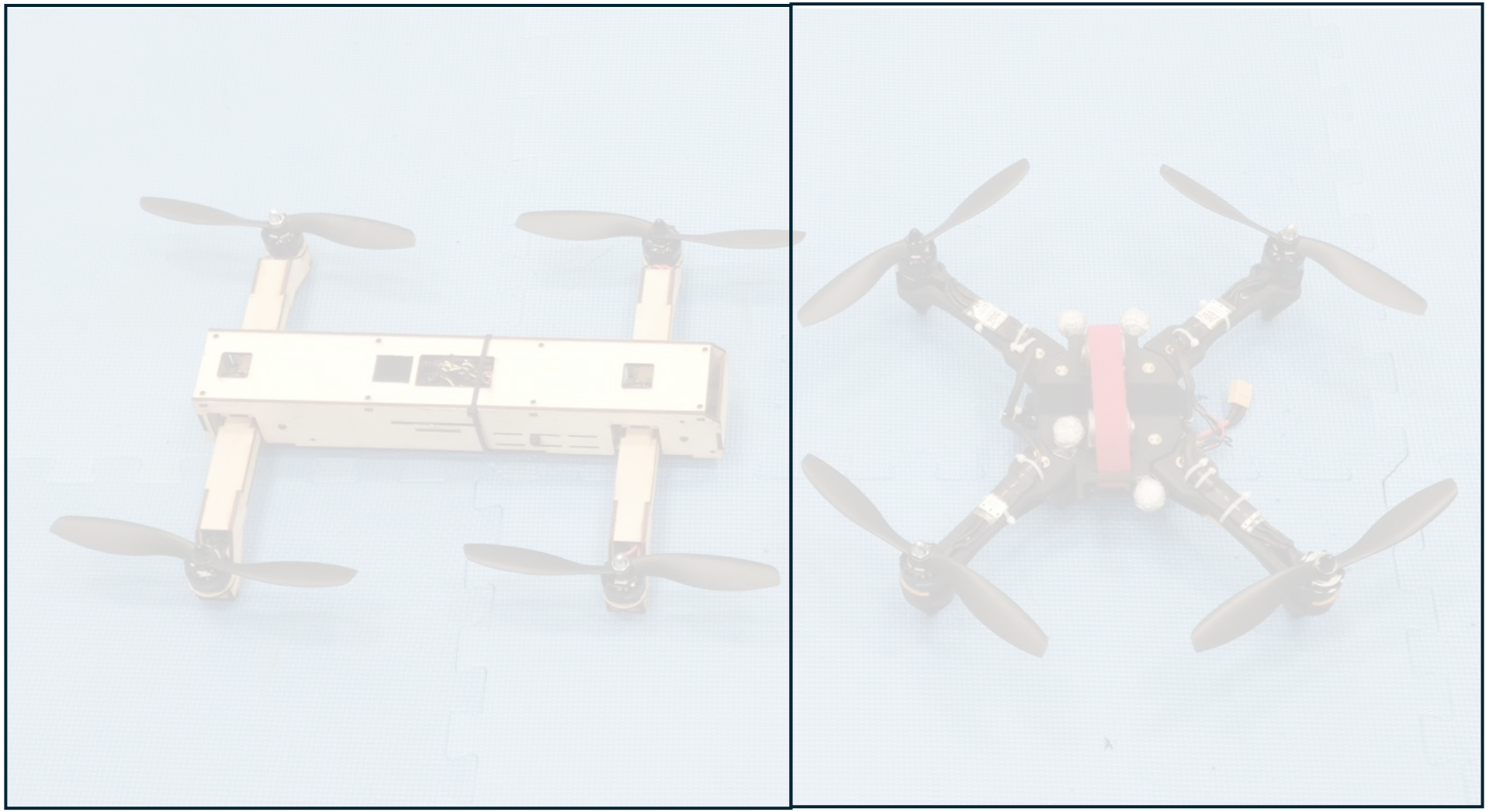
- **Interesting (but problem-specific) finding.** Physics-based randomization enables generalization beyond the training range.



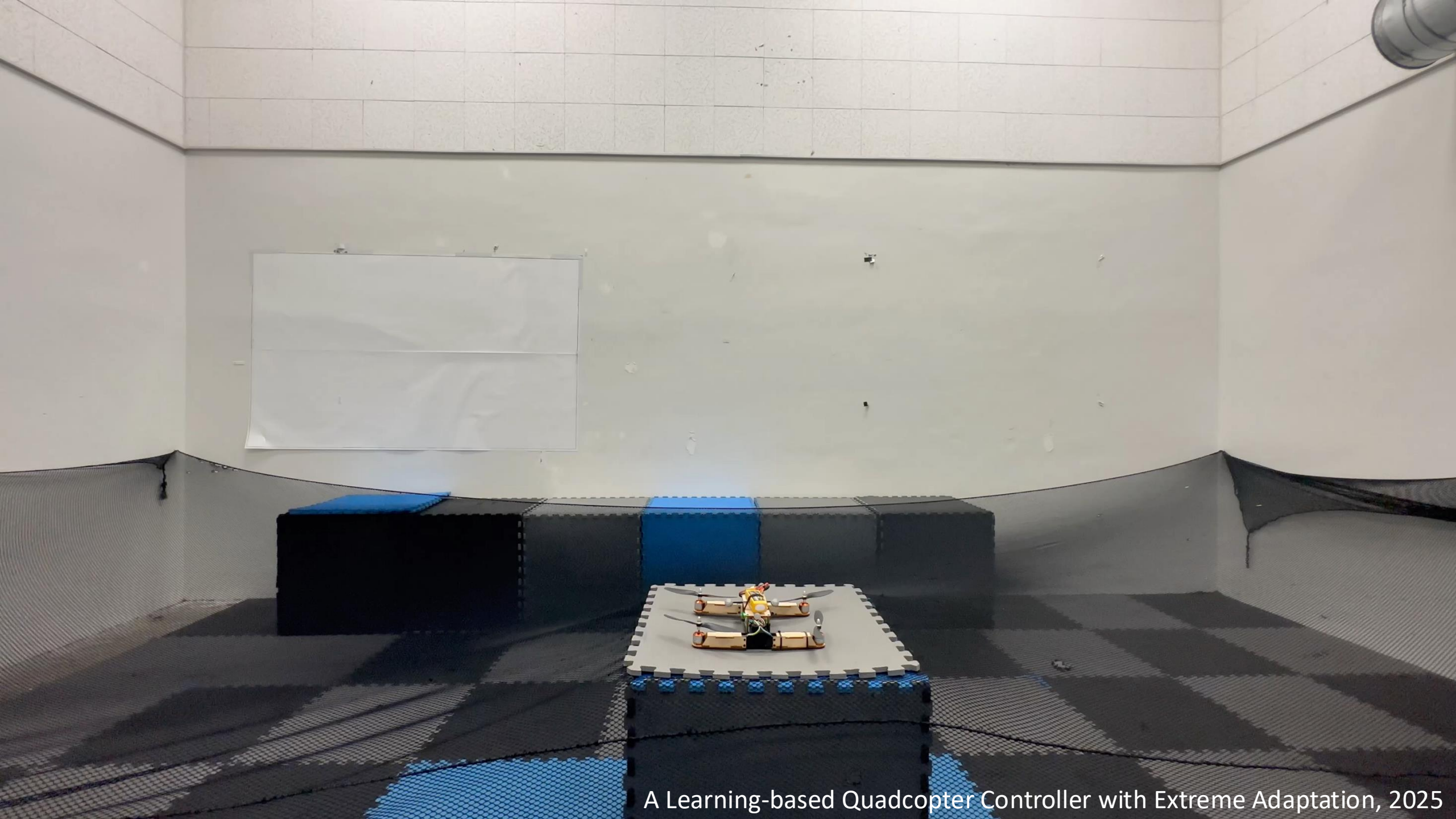


A Learning-based Quadcopter Controller with Extreme Adaptation, 2025

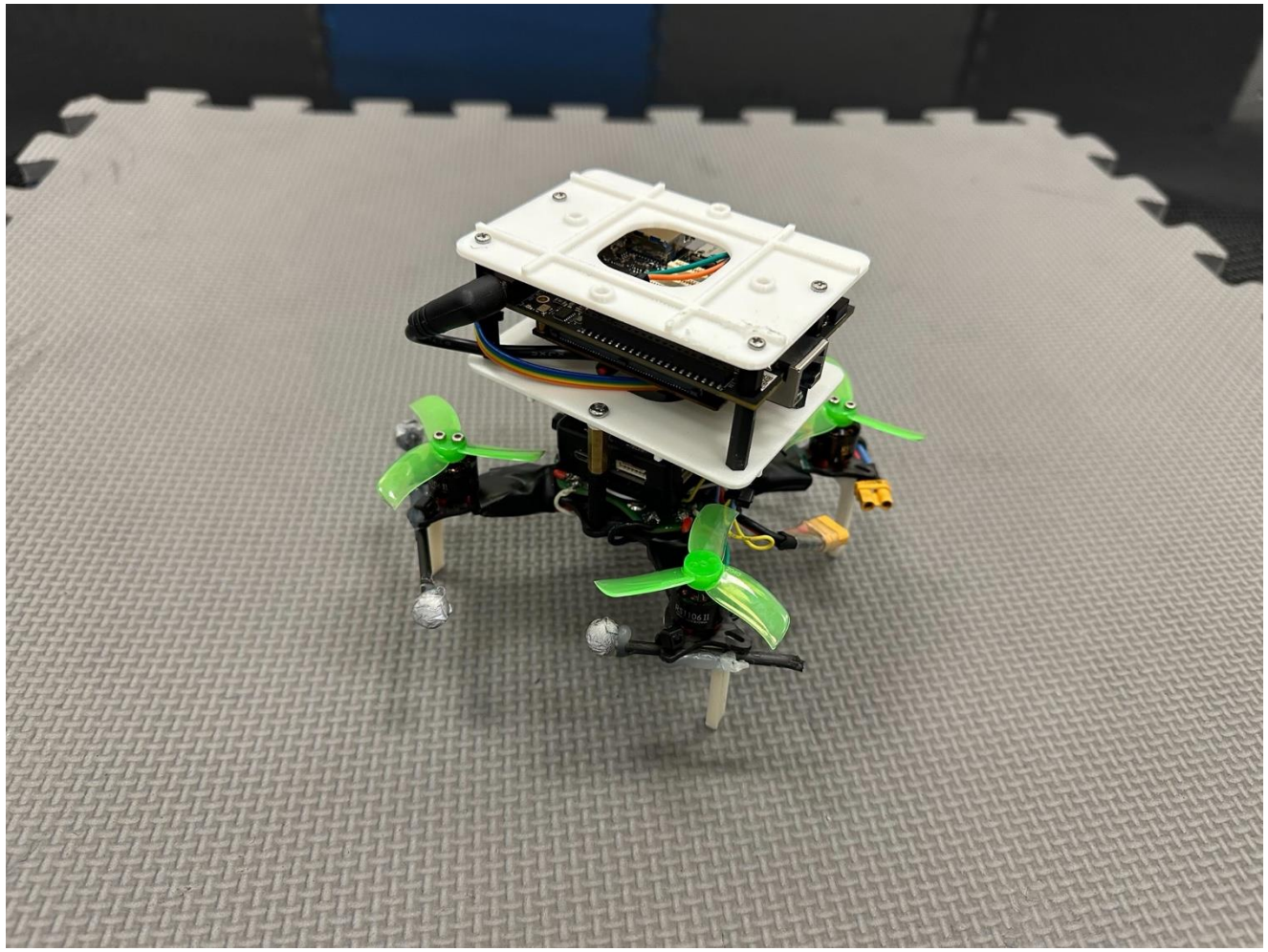
Flying Different Morphologies



QUaRTM: A Quadcopter with Unactuated Rotor Tilting Mechanism Capable of Faster, More Agile, and More Efficient Flight, Jerry Tang, Karan P. Jain, and Mark W. Mueller


A Learning-based Quadcopter Controller with Extreme Adaptation, 2025

Flying Different Morphologies



- ~30% mass above the propellers
- 3X larger x-z inertia than a normal drone.

A Learning-based Quadcopter Controller with Extreme Adaptation, 2025

A Learning-based Quadcopter Controller with Extreme Adaptation, 2025

How to pick the parameters distribution?

- Physics-based: Use prior knowledge about the system to generate parameter combinations that are physically plausible.

- Limitations?

- Unclear how to design these relationships for complex systems (e.g., a dexterous hand).
- Unclear how to apply this for sensory and semantic randomization.

Why Does DR Work?

- **Popular Justification**: We randomize parameters with the hope that the real world is in the simplex of the parameters distribution ranges.

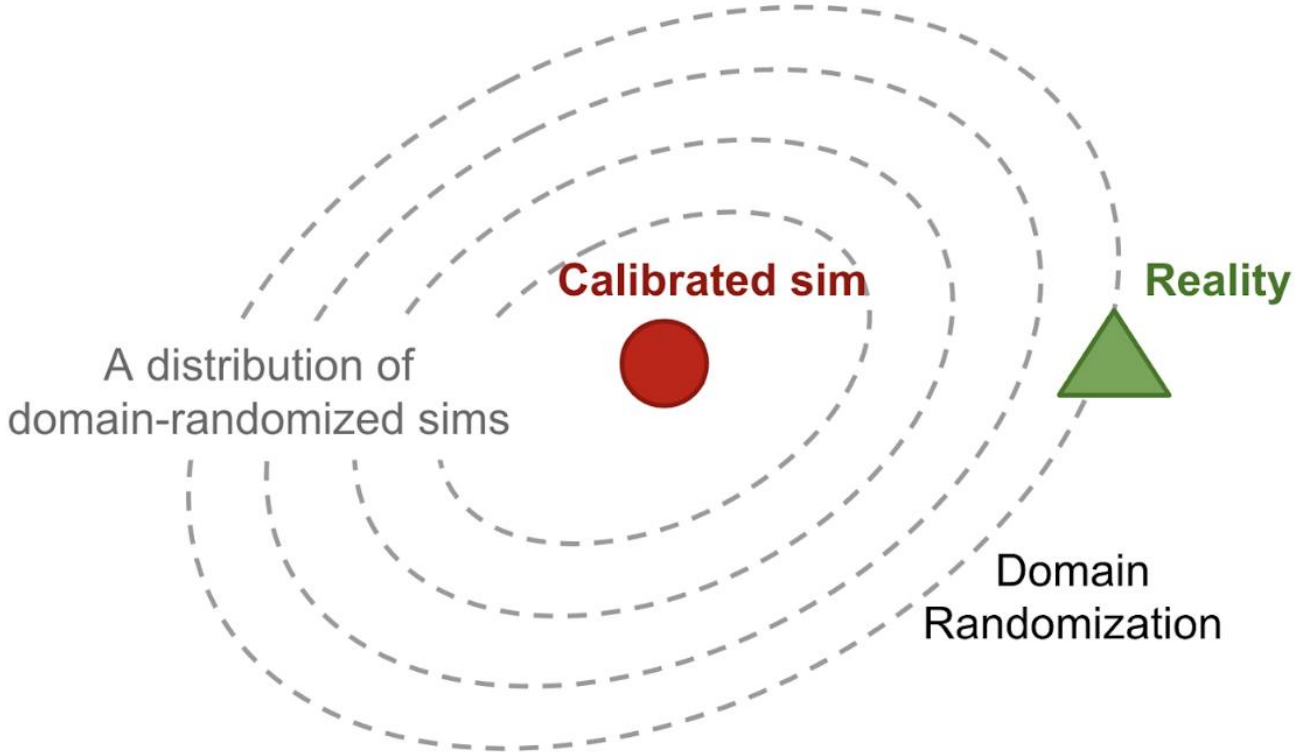


Image source: https://lilianweng.github.io/posts/2019-05-05-domain-randomization/

Why I think this is wrong (disclaimer: opinion not a fact)

- Even if you randomize everything extensively, a simulator might still not capture the true dynamics due to its inner workings (numerical integration, collision handling, force computation).

- The randomization ranges required might be so large that the probability of hitting a neighborhood of the real world is almost zero.

- It can be true at best for dynamics and sensory randomization, since they are physical parameters with a ground truth. What about the semantics randomization?

Why does domain randomization work? (Disclaimer: Opinion)

- Let's go back to the optimization objective of domain randomization

$$\theta^* = argmax_\theta \, \mathbb{E}_{e \sim D}[\, C(\theta, e)]$$

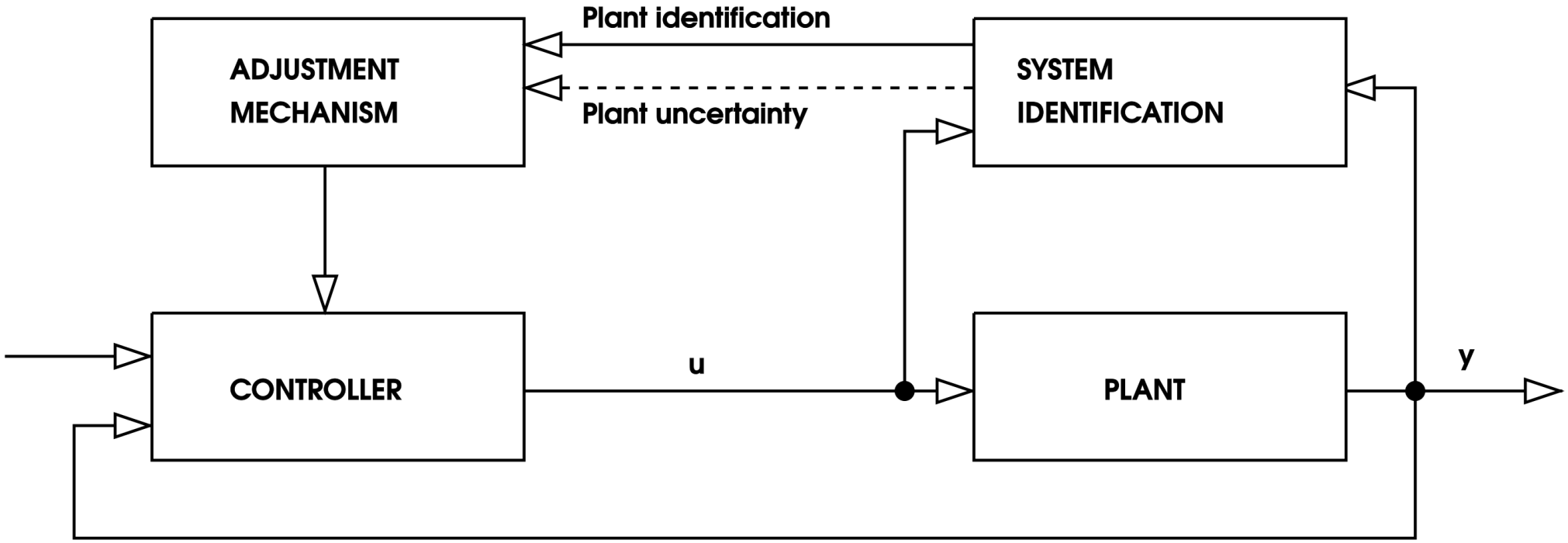
- A good solution to this problem is learning to quickly learn/adapt.
- For example, the agent can learn how to do quick estimation of parameters and learn the relation between parameters and actions (e.g., the larger the mass, the larger the force).
- This does not require that the real world is in the simplex of the parameter ranges.
- Can we design algorithms to bias the agent to learn this behavior?

Categories of Sim2Real Algorithms

- Domain Randomization
  - Naïve
  - Adaptive
  - Physics-based
- System Identification
  - Explicit System Identification (Real-to-Sim)
  - Implicit System Identification (Adaptation)
- Abstractions-based
  - Visual Abstractions
  - Control Abstractions

System Identification

- **Key Idea**: Identify the system's task-relevant parameters from (possibly limited) on-policy experience.
- Use the estimated parameters to train/improve the policy.



MODEL IDENTIFICATION ADAPTIVE CONTROL (MIAC)

Source: Wikipedia

System Identification

Mainly two types:

- **Explicit identification (Real-to-Sim)**: find the actual parameters.
- **Implicit identification (Adaptation):** find a compressed representation of the parameters.
- The two types are not mutually exclusive and can be used together.

Explicit System Identification

- The process of building a mathematical model of a dynamical system using **measured input and output data**.

- **Input:** Actions (torques, joint angles, end-effector position, etc.).

- **Output:** The measured response of the robot (e.g., velocity, observations, etc.).

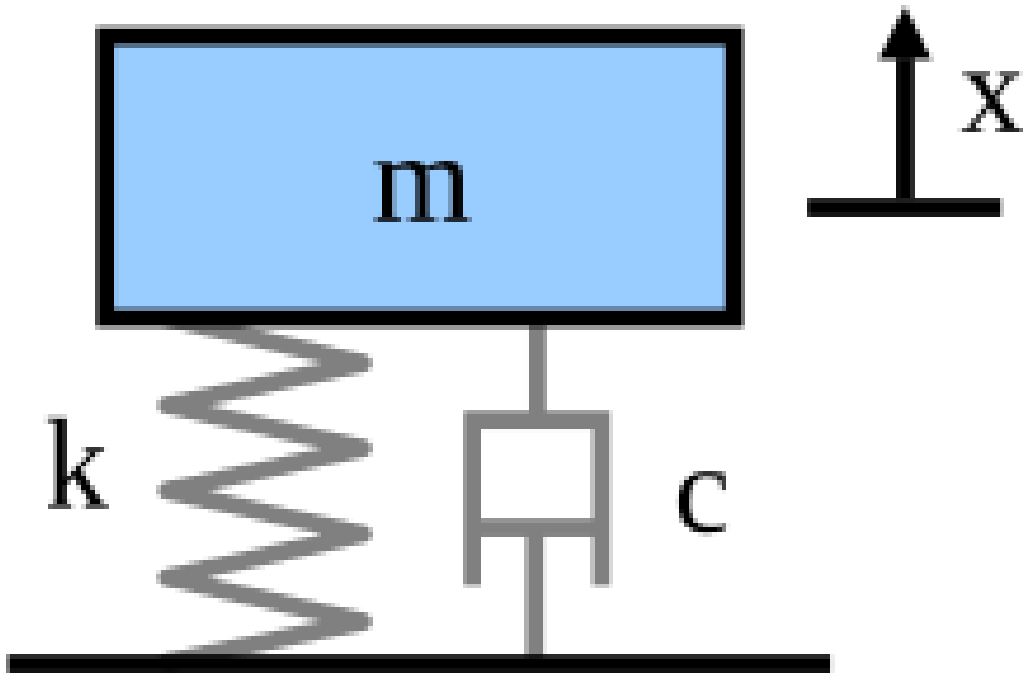
- **Goal:** To estimate the unknown parameters in the governing equations.

- **Example**: find M, C in $M\ \ddot{q}\ +\ C\ \dot{q}\ +\ G\ =\tau$.

The 5-step procedure

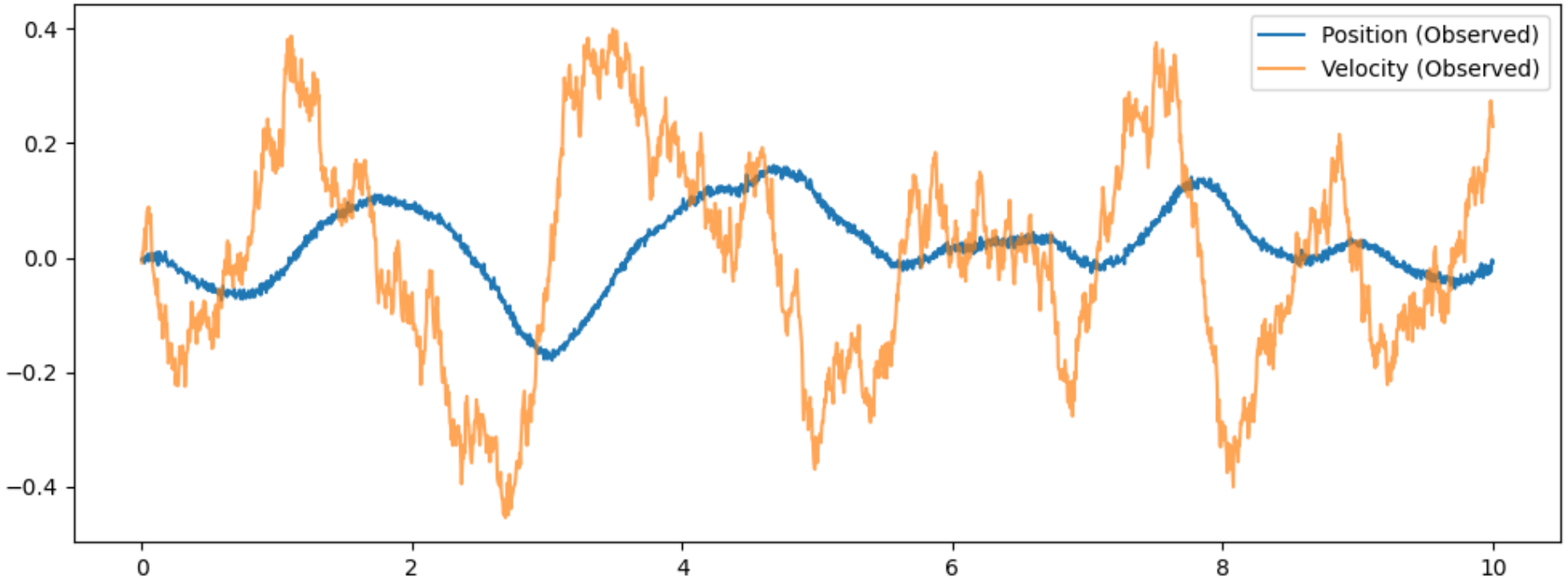
1. **Experiment design**
   - Choose an “exciting” signal that forces the robot to “reveal” its dynamics.
2. **Data Collection**
   - Execute the inputs on the physical hardware and record the sensor data.
3. **Model Structure Selection**
   - Decide on the mathematical form used to model the system.
4. **Model Selection**
   - Use optimization algorithms to find parameter values that minimize the difference between the model's output and the real data.
5. **Model Validation**
   - Test the model against a *new* set of data (the validation set). If the "fit" is poor, return to step 1 or 3.

Example: Estimating the parameters of a smd system



Example: Estimating the parameters of a smd system

- Experiment design:
  - random force
- Data collection:
  - Forward propagate a smd system with fixed parameters

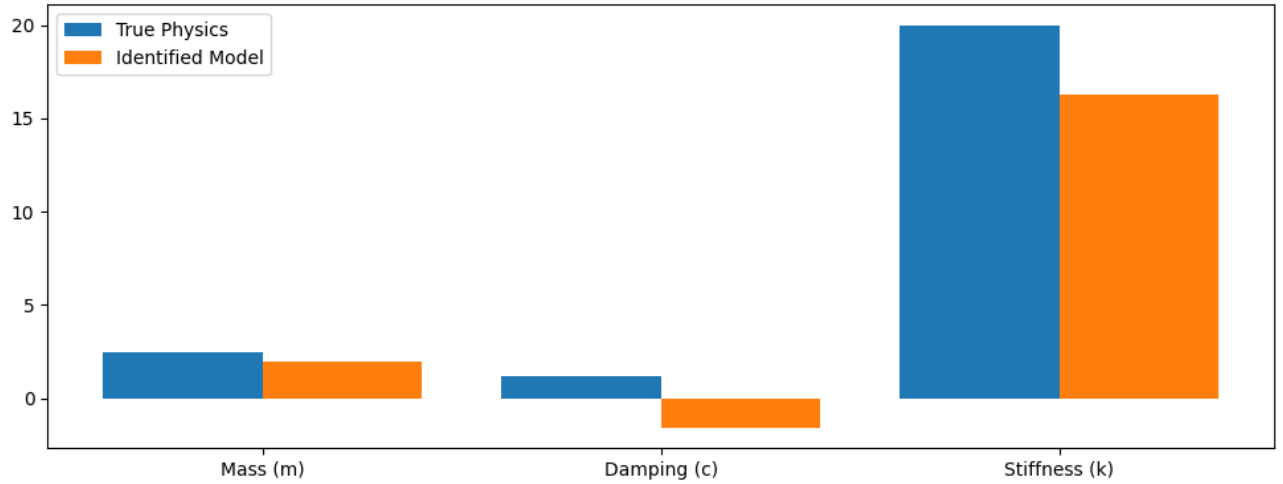


Example: Estimating the parameters of a smd system

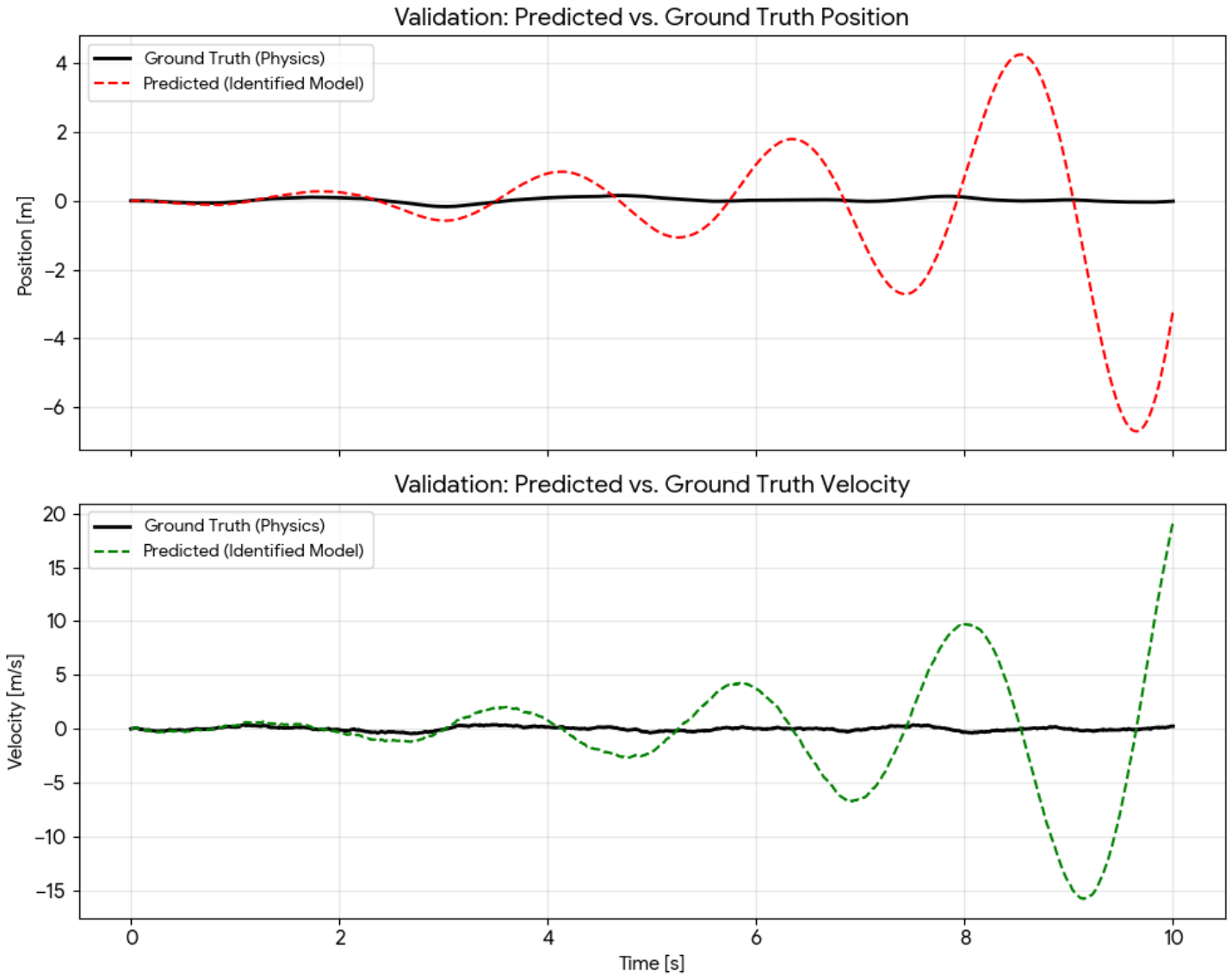
- Experiment design:
  - random force
- Data collection:
  - Forward propagate a smd system with fixed parameters
- Model Structure Selection:
  - $F = ma + cv + kx$

Example: Estimating the parameters of a smd system

- Experiment design:
  - random force
- Data collection:
  - Forward propagate a smd system with fixed parameters
- Model Structure Selection:
  - $F = ma + cv + kx$
- Model Selection using least squares:



Final Step: Model Validation



Fix this at home!

Challenges of Explicit Identification

- Highly dependent on the mathematical model that is selected
  - Hard-to-model nonlinearities (e.g., motor backlash) could lead to poor estimation accuracy.

- Highly dependent on the quality of the sensor observations
  - The larger the noise, the worse the estimates.

- If the training data isn't "exciting" enough, the model will "overfit”
  - This will probably result in a bad policy.

Explicit System Identification

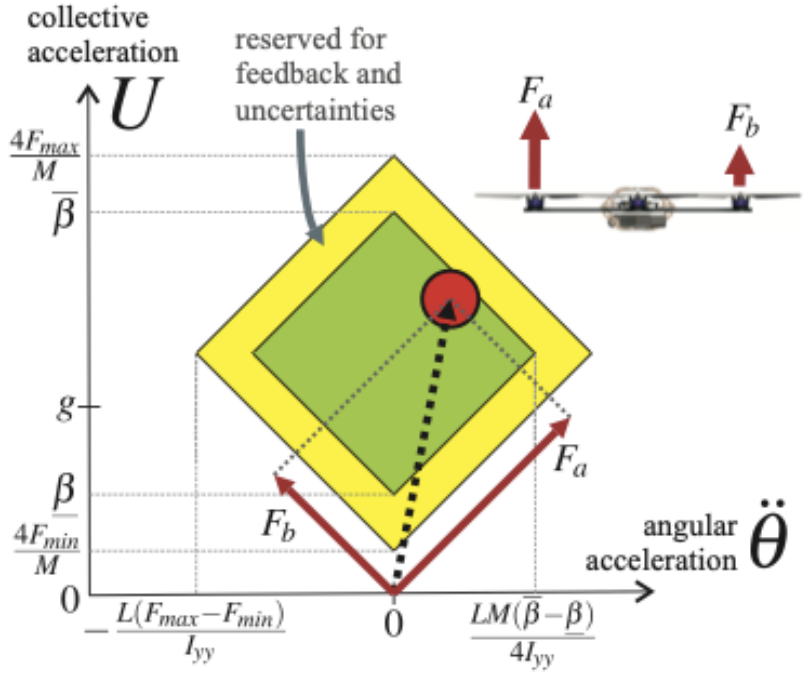
A Simple Learning Strategy for High-Speed Quadrocopter Multi-Flips

Sergei Lupashin, Angela Schöllig, Michael Sherback, Raffaello D'Andrea

ICRA, 2010

Explicit System Identification

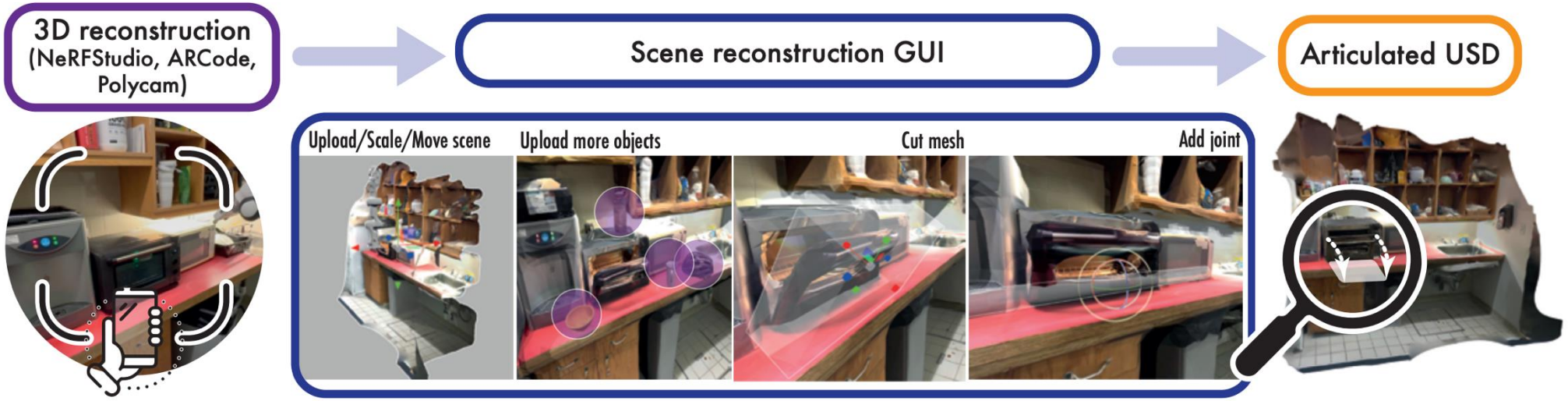
- Estimation is never perfect: uncertainty in some parameters always remains.
- To account for this, explicit identification is generally coupled with domain randomization.



A Simple Learning Strategy for High-Speed Quadcopter Multi-Flips, 2010

Explicit System Identification: **A Modern View**

Modern take on the problem: estimate not only physical parameters, but the whole scene.



Reconciling Reality through Simulation: A Real-to-Sim-to-Real Approach for Robust Manipulation, 2024

Explicit System Identification: **A Modern View**

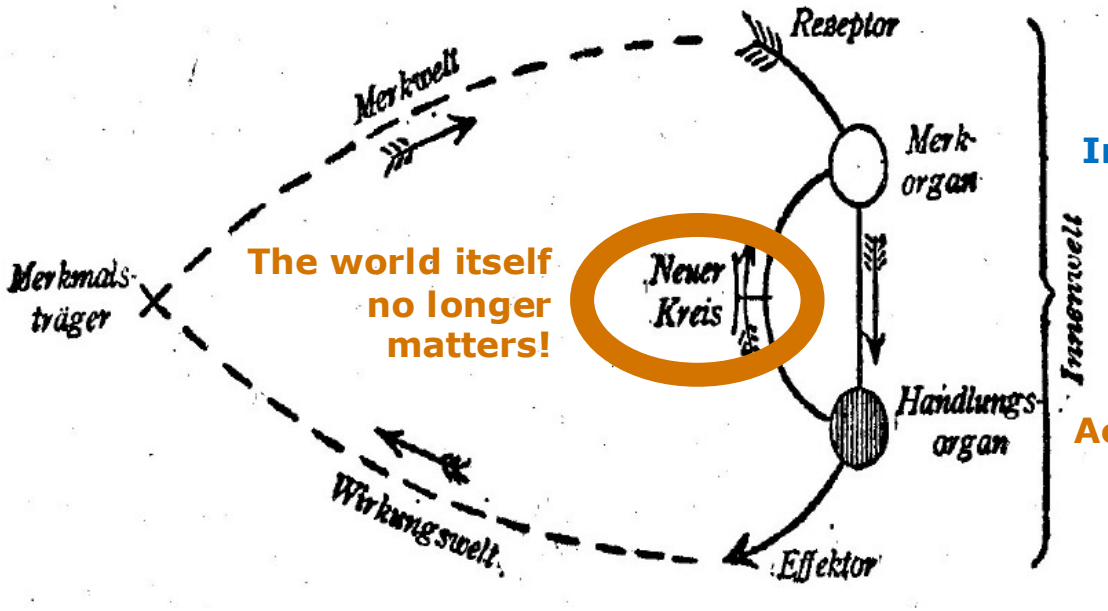
Limitations of scene-level real-to-sim?

- The more complex the scene, the slower the simulation.
- Articulation of objects is non-trivial when done for more than a handful of objects (particularly hard for deformable objects).
- Reconstruction artifacts might impact the behavior of the policy. Policies are generally finetuned, not trained from scratch.

However, we don't (yet) have better methods to cope with the semantic sim2real gap!

Implicit System Identification

- **Key question:** Do we need to find the ground-truth system parameters to train a good policy?



Feedback about the action

The world itself no longer matters!

Input from the world

Action affects the world

Implicit System Identification

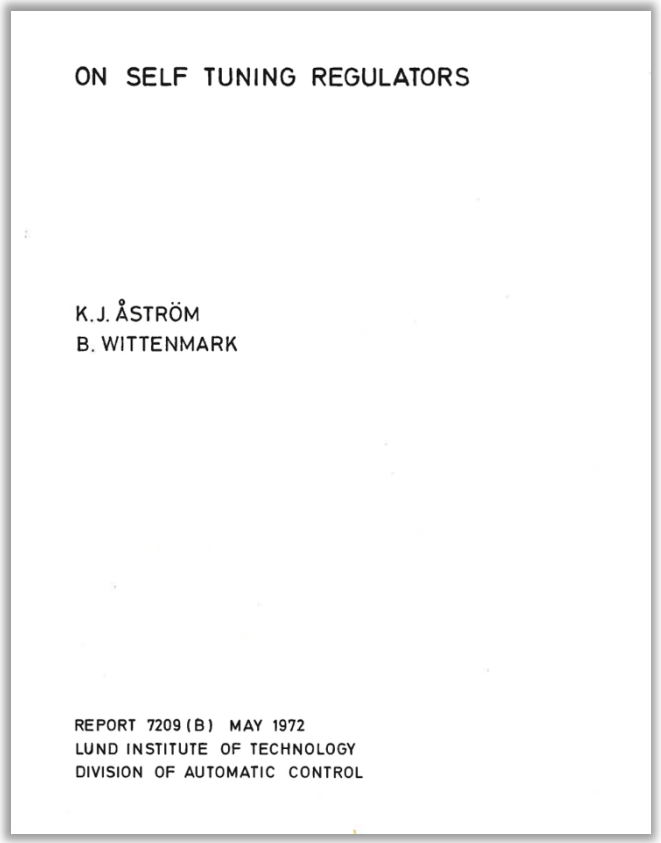
- **Key insights**

- ***Parameters can often be uniquely identified.***
- Example: payload or weak motor?

- ***Parameters that cannot be uniquely identified are not necessary for the downstream task.***
- Example: Flying on a vertical line does not allow you to estimate a drone's inertia. But you don't need the inertia to fly in a vertical line!

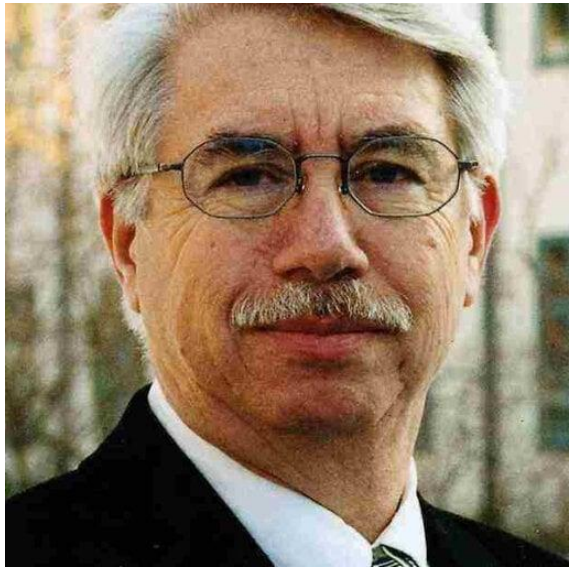
Implicit System Identification

- The origins of these techniques are in the adaptive control literature for linear systems.


ON SELF TUNING REGULATORS

K.J. ÅSTRÖM
B. WITTENMARK

REPORT 7209 (B) MAY 1972
LUND INSTITUTE OF TECHNOLOGY
DIVISION OF AUTOMATIC CONTROL



Implicit Adaptive Control

- More recently adapted to learning-based approaches and sim2real.

- **Approach:** learn a compressed parameter representation.
- **Assumption**: compressibility implies predictability.

Initially proposed by:

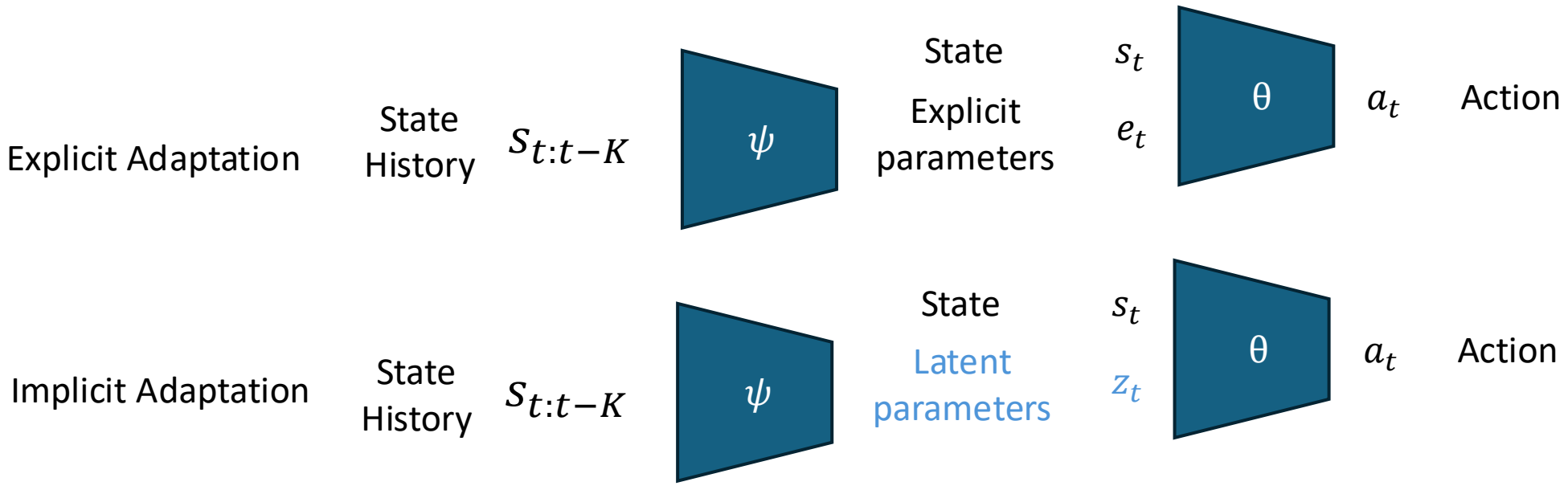
1. Learning Quadrupedal Locomotion over Challenging Terrain, Lee et al., 2020

2. Rapid Motor Adaptation, Kumar et al., 2021

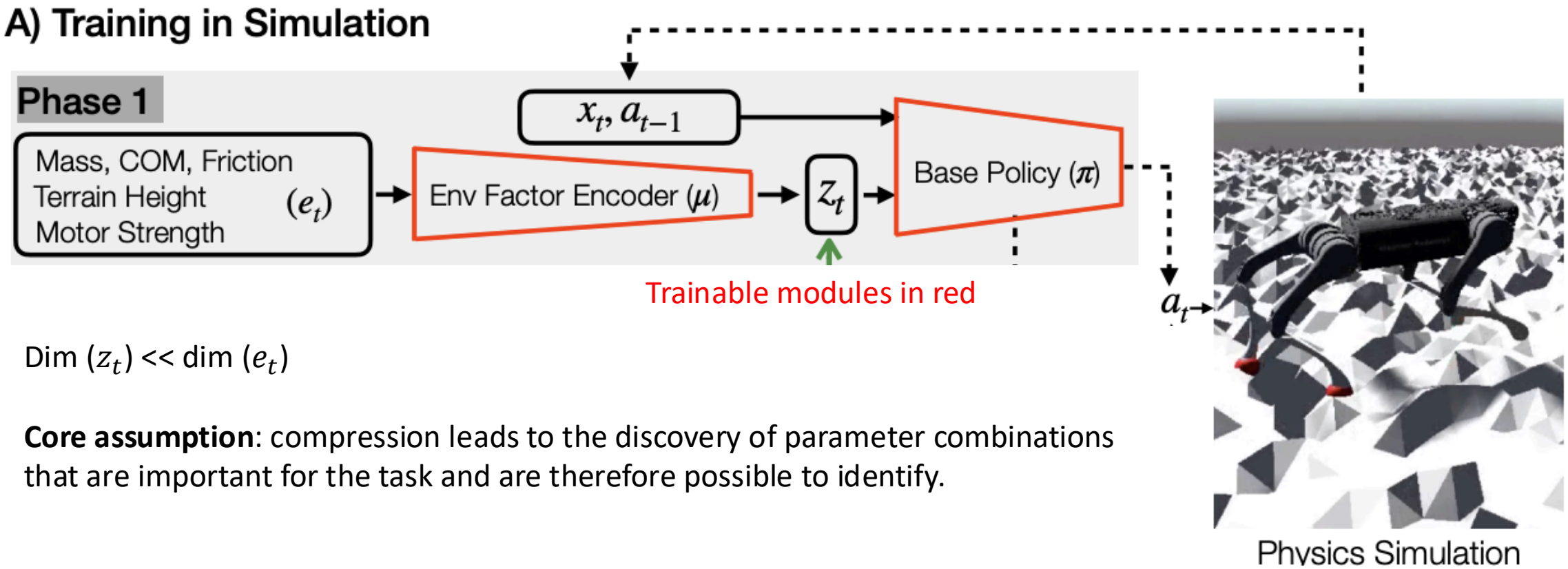
Implicit Adaptive Control

- More recently adapted to learning-based approaches and sim2real.

- **Approach:** learn a compressed parameter representation.
- **Assumption**: compressibility implies predictability.



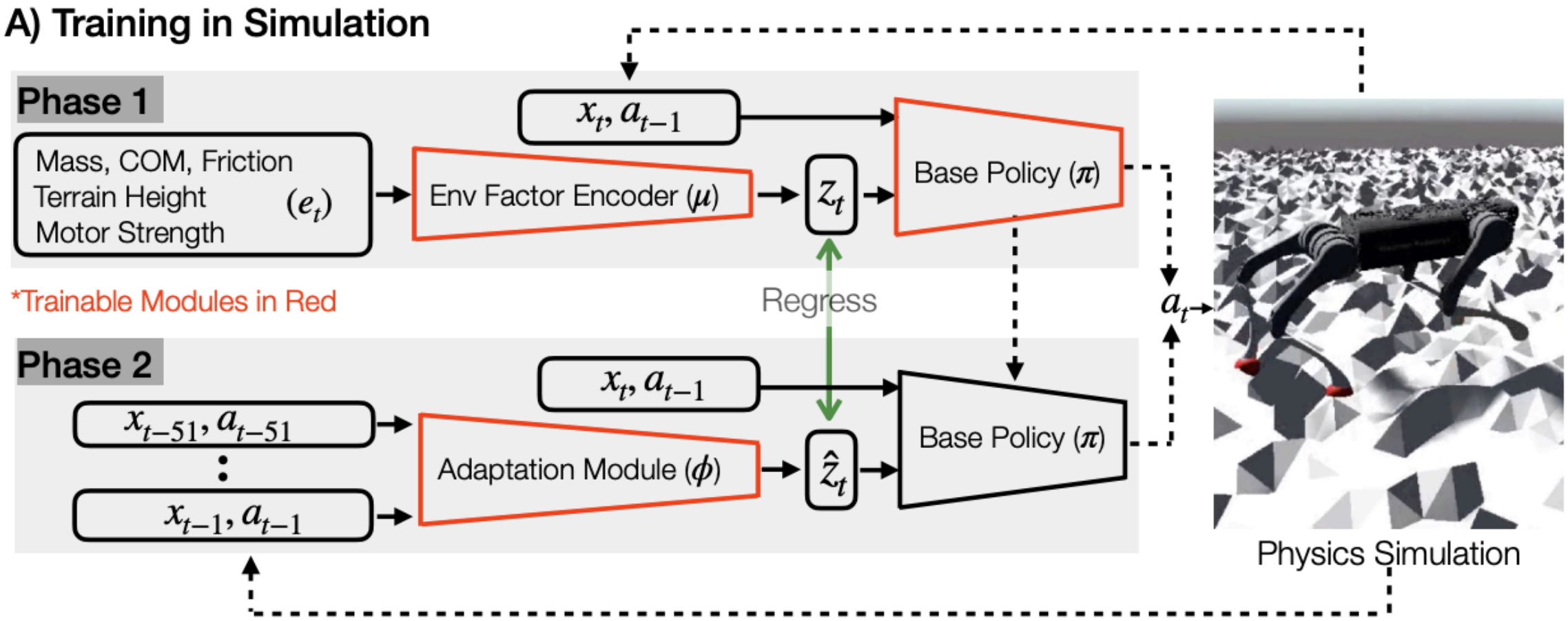
Rapid Motor Adaptation



Trainable modules in red

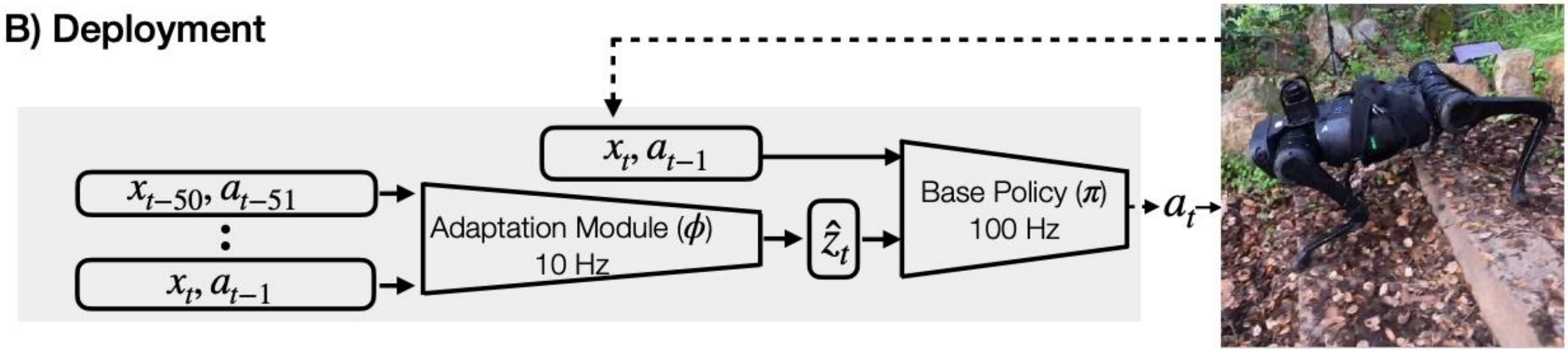
- Dim (z_t) << dim (e_t)
- **Core assumption**: compression leads to the discovery of parameter combinations that are important for the task and are therefore possible to identify.

Rapid Motor Adaptation



- The history of previous actions and states is used for predicting the latent representation of parameters.
- The action policy is kept frozen (in some implementations)

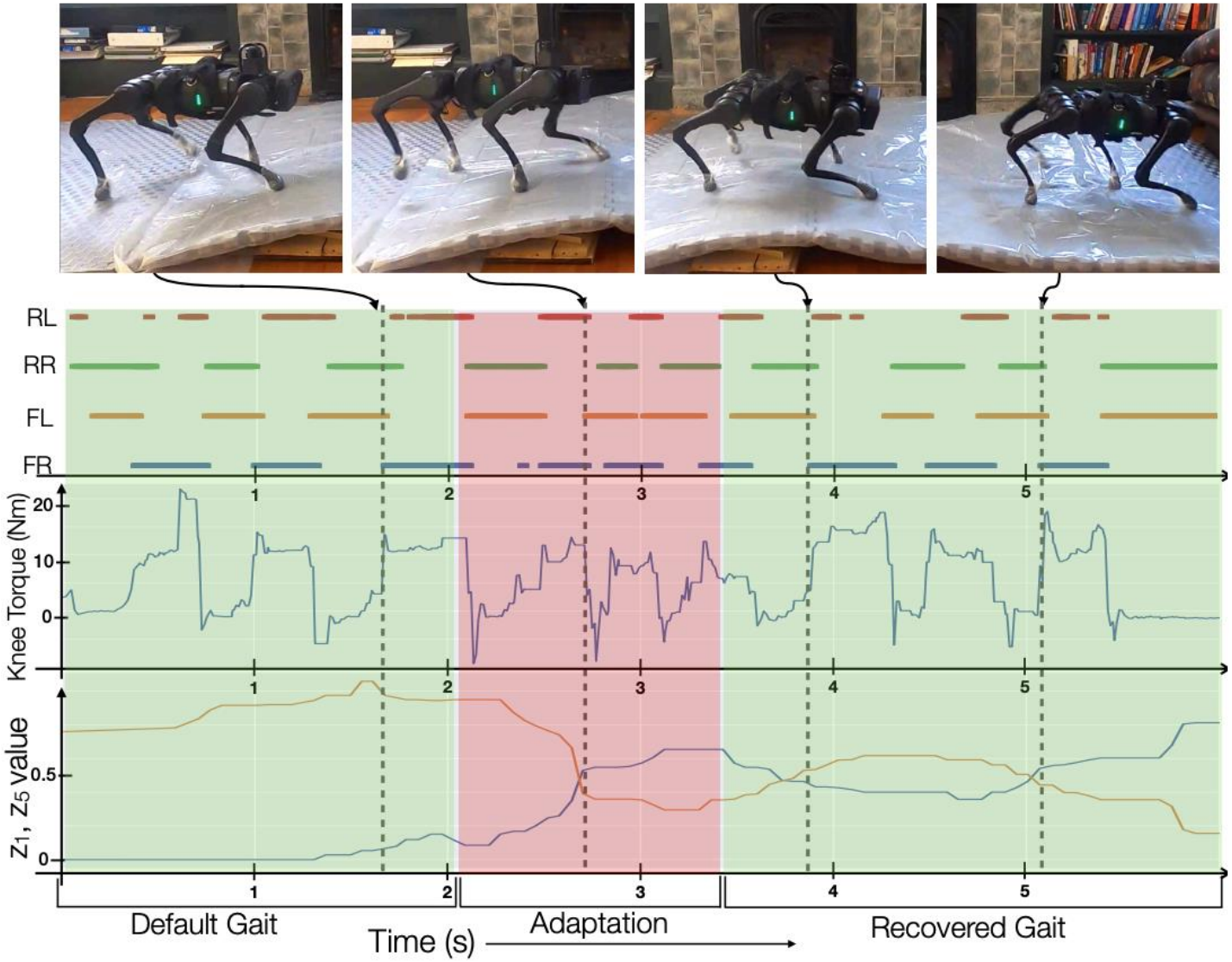
Rapid Motor Adaptation



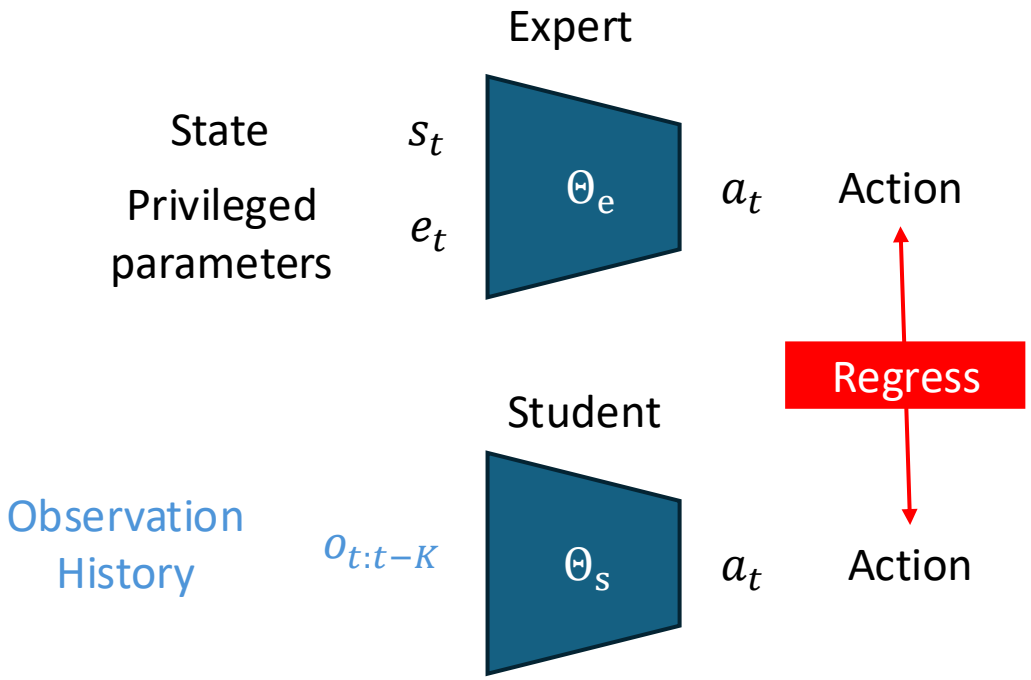
- The adaptation module and policy can be deployed zero-shot on the robot.
- Strong performance in environments not modeled at training time (e.g., sand, stairs, foliage)

Quick Adaptation

Quick Adaptation



Relation to Prior Work: Imitation of Privileged Policy



Learning by cheating, Chen et al, 2019

The approach has been applied to many control problems

- Parkour
  - Examples:
    - Parkour in the wild: Learning a general and extensible agile locomotion policy using multi-expert distillation and RL Fine-tuning, Rudin et al. 2025
    - Extreme Parkour with Legged Robots, Cheng et al., 2024
    - Robot Parkour Learning, Zhuang et al., 2023
- Vision-based locomotion
  - Examples:
    - Legged Locomotion in Challenging Terrains using Egocentric Vision, Agarwal et al, 2022
    - Learning robust perceptive locomotion for quadrupedal robots in the wild, Miki et al, 2022
- General takeaway:
  - The more challenging the task is, the more tricks are required (several RL experts, curriculum on contacts models, tailored perception systems).
  - The approach does not simply scale.

Limitations and How to (partially) address them

- The compression objective could not be sufficient to achieve the goal of predictability.
  - Add an extra objective during phase I training which focuses on predictability
  - Example:
    - Deep Whole-Body Control: Learning a Unified Policy for Manipulation and Locomotion, Fu et al. 2022.
- The base policy should potentially change its behavior when it does not observe the ground truth parameters.
  - Fine-tune the base policy together with the adaptation module with RL.
  - Examples:
    - Adapting Rapid Motor Adaptation for Bipedal Robots. Kumar et al., 2022

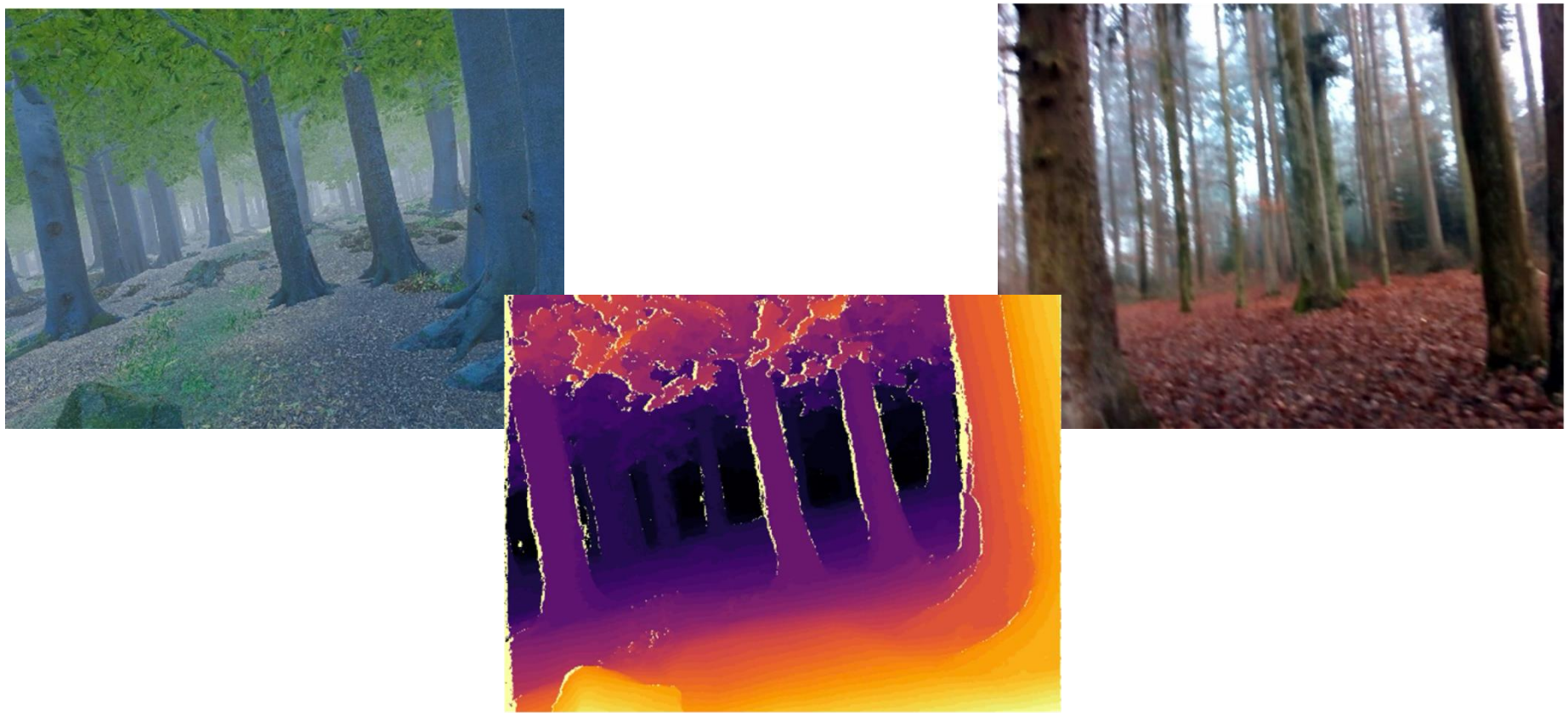
- A two-stage procedure is cumbersome. Errors can compound between phases.
  - Recent work seems to prefer the reward/observation engineering route

Categories of Sim2Real Algorithms

- Domain Randomization
  - Naïve
  - Adaptive
  - Physics-based
- Adaptive Control
  - Explicit System Identification
  - Implicit System Identification
- Abstractions-based
  - Visual Abstractions
  - Control Abstractions

Abstraction-based Sim2Real

General idea: train the policy on *curated* sensory/control abstractions that reduce the gap between simulation and real world.



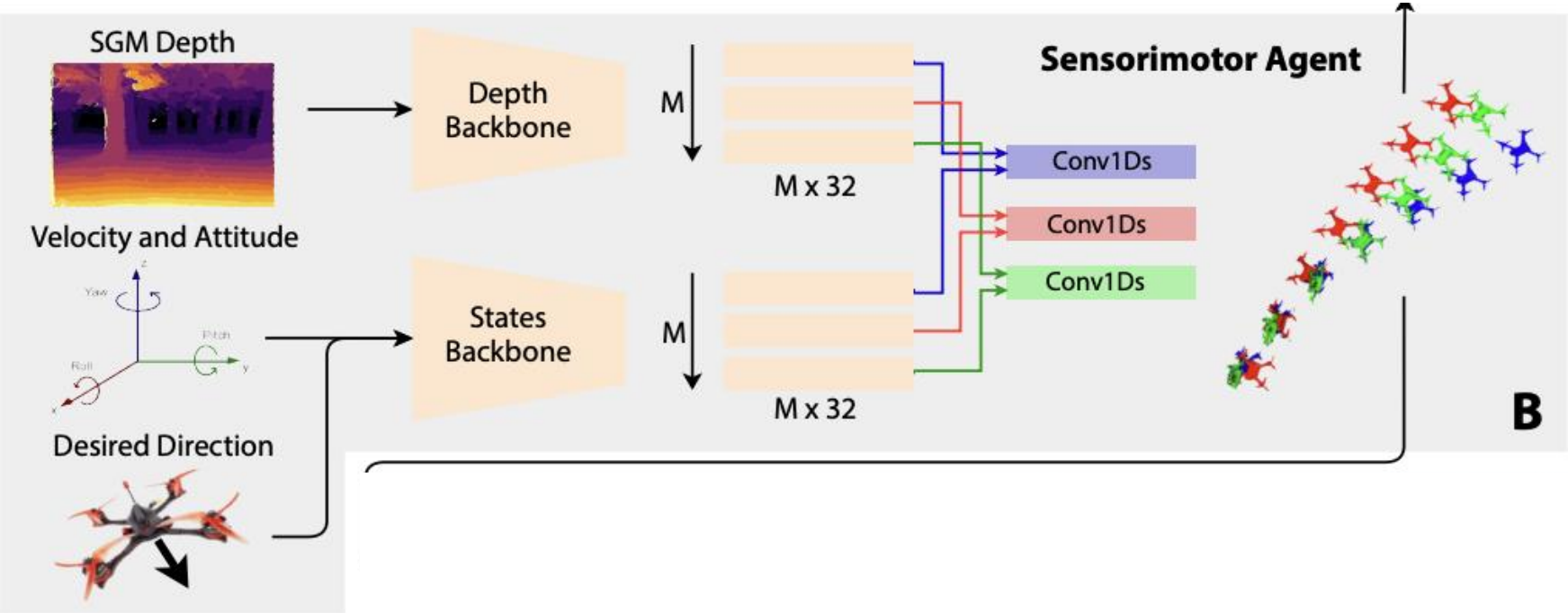
Examples of Sensory Abstractions

- Gate Detection in drone racing
- Pose estimation and mapping for navigation
- 6D Object Pose detection for manipulation
- Many others (each problem can have its own observation engineering)

Control Abstractions

- Low-level controllers (e.g., MPC) can track trajectories or joint positions, so that the policy can focus on higher-level control.
  - Paradoxically, this could potentially lead to an increased sim2real gap if the low-level controller does not behave similarly in simulation and the real world.
  - Randomization becomes harder to do (need to make the low-level adaptive)

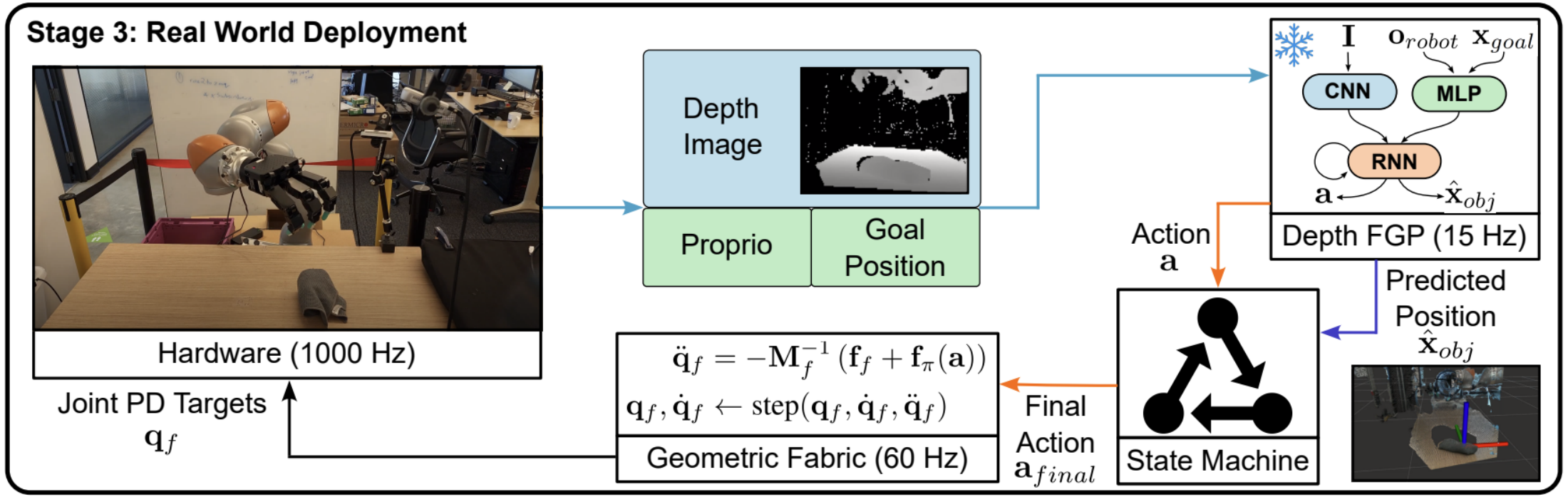
An example: High-Speed Obstacle Avoidance



Learning High-Speed Flight in the Wild, Loquercio et al., 2021

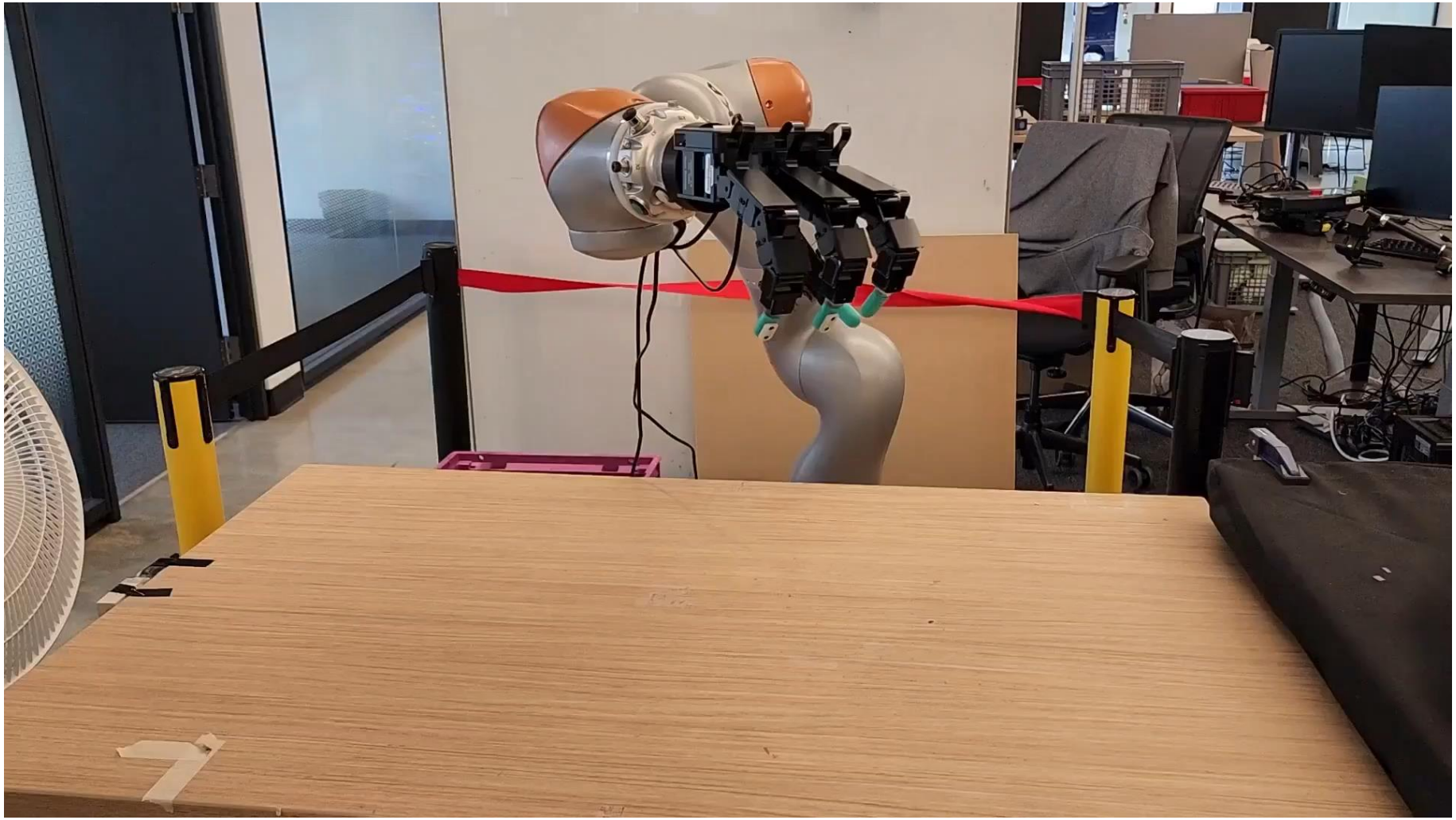
Loquercio et. al.,
Science Robotics, 2021

An example: Dexterous Manipulation



DextrAH-RGB: Visuomotor Policies to Grasp Anything with Dexterous Hands, Singh et al., 2025
DextrAH-G: Pixels-to-Action Dexterous Arm-Hand Grasping with Geometric Fabrics, Lum et al., 2024

An example: Dexterous Manipulation



DextrAH-RGB: Visuomotor Policies to Grasp Anything with Dexterous Hands, Singh et al., 2025
DextrAH-G: Pixels-to-Action Dexterous Arm-Hand Grasping with Geometric Fabrics, Lum et al., 2024

Abstraction-based Sim2Real

- **General idea:** train the policy on *curated* sensory/control abstractions that reduce the gap between simulation and real world.

- A neat way to introduce inductive bias in the policy learning process.

- Limitations?
  - Abstractions require modularity, which comes with several disadvantages (e.g., compounding errors, latency, computing the abstraction might be harder than the task itself).
  - Abstractions are mostly task-specific.

Categories of Sim2Real Algorithms

- Domain Randomization
  - Naïve
  - Adaptive
  - Physics-based
- System Identification
  - Explicit System Identification
  - Implicit System Identification
- Abstractions-based
  - Visual Abstractions
  - Control Abstractions

Most papers use a mix of these techniques. But some applications require care.

Tips and Tricks to Make Your Sim2Real Pipeline Work

The Steps of a Sim2Real Pipeline

- Step 1: Decide on your abstractions!
  - **Sensory**
    - Which sensory inputs can I get in the real world?
    - What is the noise associated with these inputs?
    - Are these sensory inputs fast or slow to simulate? (Point clouds are very fast, but depth/RGB images are slow)
  - **Control**
    - At which level should my policy operate? (motor commands, joint positions, velocity commands, pose commands, etc.)
    - How well can you simulate the control hierarchy on which the policy will rely? (example: the Franka manipulator).

The Steps of a Sim2Real Pipeline

- Step 2: Decide on your strategy for policy training

  - **Single Step?**
    - Mostly works if your policy is reactive and/or sensor simulation is fast.
    - Well-suited to policy learning via imitation.
    - More challenging for policy learning with RL: Might require very careful reward engineering.
    - Core advantage: the policy can be directly deployed on hardware.

  - **Two (or more) Steps? (Privileged Training + Distillation via Imitation)**
    - Train a privileged policy with ground-truth information (e.g., location of obstacles)
    - Distill the privileges policy into another policy with sensor observation (generally called the teacher-student paradigm)
    - Most common for policy learning with RL when you have some sensory inputs or control abstraction that are slow to simulate (e.g., images).

The Steps of a Sim2Real Pipeline

- Step 3: Decide on your randomization/adaptation strategy

  - **Randomize and conquer**
    - Identify the parameters that are uncertain and might require randomization
    - Identify suitable randomization ranges
    - The less you randomize, the faster and more effective policy training will be.
    - The less you randomize, the lower the probability of effective sim2real.

  - **Shall I use an adaptation module?**
    - If yes, you are forced to a two-step training procedure (see previous slide). However, it often helps performance.
    - If you're doing a two-step procedure anyway because your sensor simulation is slow, it is worth doing it.

The Steps of a Sim2Real Pipeline

- Step 4: Design your environment

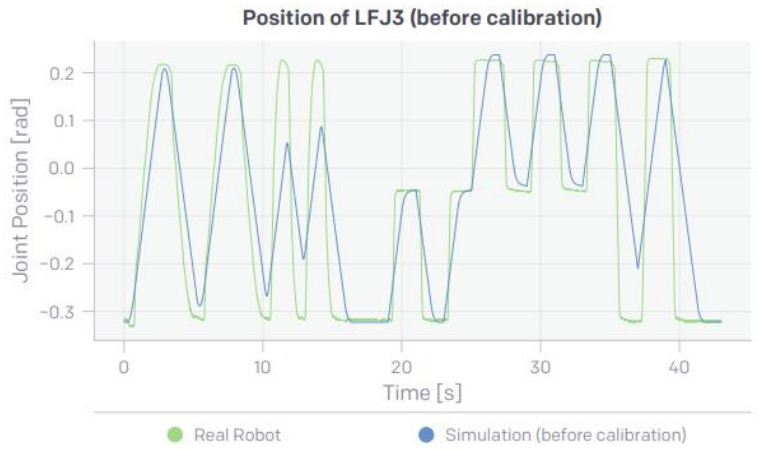
  - **One of the most complicated parts**
    - What objects do you need in the scene to accomplish the task?
    - Is geometry sufficient, or do you also need visual realism?
    - Scene randomization is important! The axes of randomization depend on your task.

  - **Do not overthink it**
    - Often, the environments in the simulation and the real world do not need to match exactly.
    - The more complex the scene, the slower the simulation will be.
    - Start simple and increase complexity if you notice specific failures.

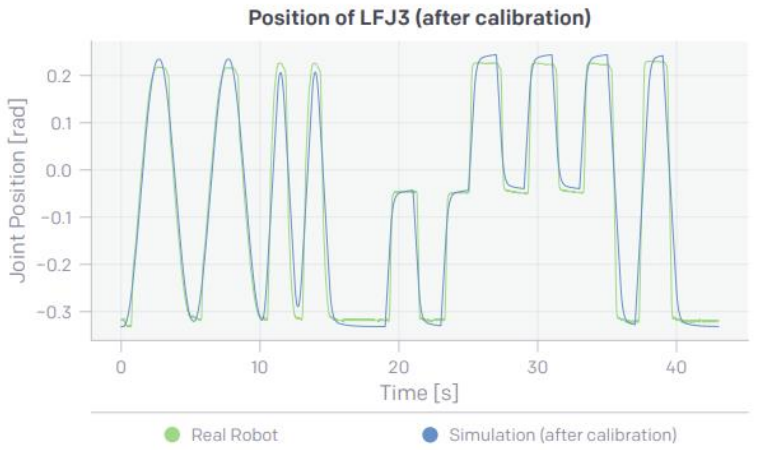
Tips to Improve Your Sim2Real Performance

Tune your control abstraction properly!

- Having the same PID gains in simulation and the real world is neither sufficient nor necessary.
- You need to match the behaviors as much as possible (the gains could potentially be different).
- Keep latency into account.



(a) Comparison against original simulation.



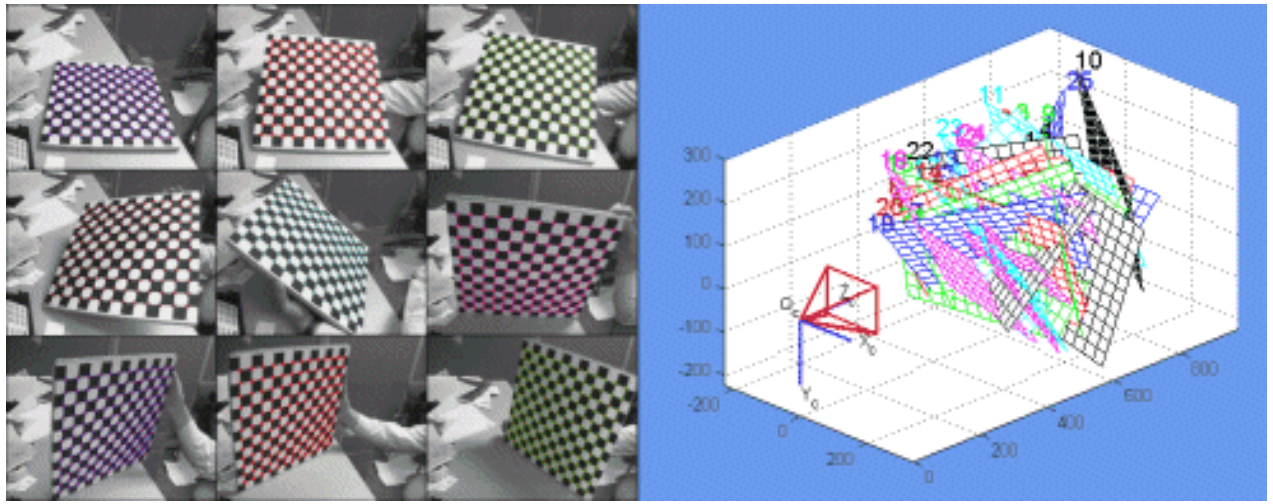
(b) Comparison against new simulation.

Solving Rubik's Cube with a Robot Hand, Akkaya et al.

Tips to Improve Your Sim2Real Performance

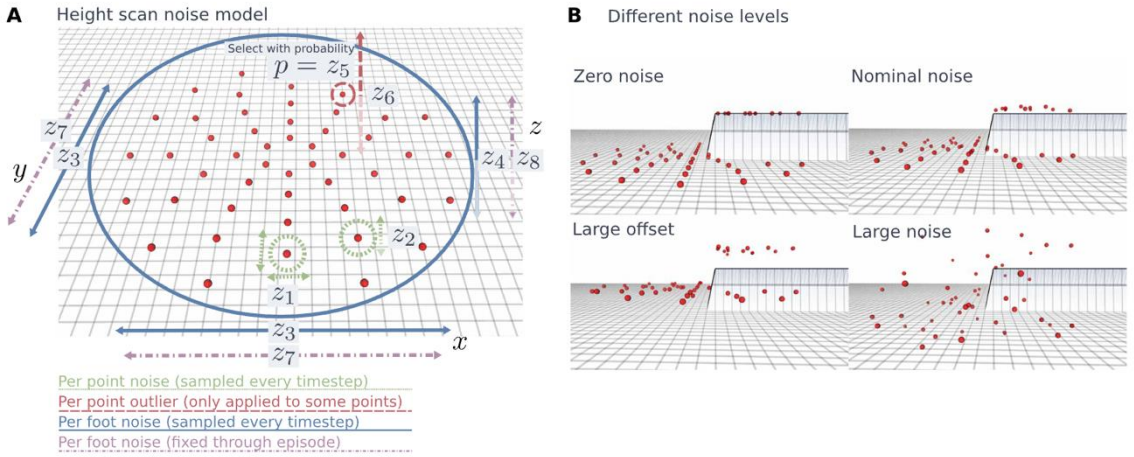
Calibrate your sensors

- Tiny errors in sensor placement can lead to vastly different observations.
- Do a good calibration (using off-the-shelf tools) and add a small randomization on top.



Tips to Improve Your Sim2Real Performance

- **Use noise models when matching the sensor-control abstractions is challenging.**



- **Train the noise models on real-world data**
  - Add a residual neural network to predict dynamics or sensor observations.
  - Examples:
    - $\mathrm{NM}^{dynamics}(\mathrm{s_t}, \mathrm{a_t}) = f_{real}(s_t, a_t) - f_{sim}(s_t, a_t)$
    - $\mathrm{NM}^{obs}(s_t) = obs_{real}(s_t) - obs_{sim}(s_t)$

Tips to Improve Your Sim2Real Performance

Evaluate often on the physical robot

- Unfortunately, often strong performance in simulation does not lead to strong performance in the real world.
  - **Corollary**: Checkpoint picking is hard; Different seeds of the same run could have vastly different performance.
  - **Sim2Sim** (e.g., Isaac to Mujoco) is often very informative in the early stage of a project.
- Do a detailed analysis of any real-world evaluation you do.
  - Did the control abstraction work as I expected?
  - Did the visual abstraction work as I expected?
  - How different is the behavior in the simulation and the real world?

High-Level Advice for Successful Sim2Real

- **Start simple, move slowly**
  - As end-to-end as possible, without complex sensor/control abstractions that are hard to match.
  - Increase the complexity of abstractions if real-world results are not as good.
  - Only add learning-based noise models if strictly necessary (they come with a lot of challenges in terms of training and generalization).
  - Don’t be afraid to mess around with the simulator’s engine (contact models, integrator, robot's physical parameters).
- **Use good engineering practices**
  - Don’t do more than one change at a time.
  - Don’t use ROS unless strictly necessary (modelling non-constant communication delays can be hard)
  - Use fast simulators, at least in early development.
  - (RL-Specific): Keep your rewards simple.

Discussion and Outlook

- Sim2Real is a promising approach for robot learning, because it enables:
  - Train safely
  - Carefully control the experience of the agent during training
  - Explore counterfactuals

- **Limitations:**
  - Not yet a clean science. It looks more like a collection of good practices
  - No good/scalable way to account for the semantic gap between simulated and real-world environments
    - **Corollary:** Environment design is often the hardest part of a sim2real project

- **Modern directions:**
  - Learning abstractions from data
  - Sim2Real from data-driven simulators
  - Continual learning of simulator, policy, and abstractions