ECE7121 Learning-based control – 2025 Fall

Safe RL / Sim2Real

INHA UNIVERSITY

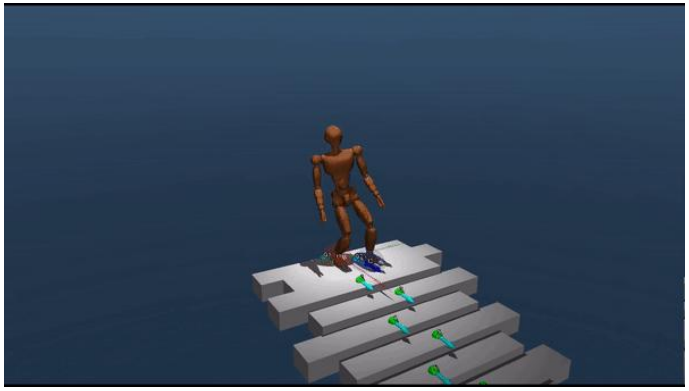
Overview

- Safe RL
  - safety
  - safe model-free RL
  - safe model-based RL

- Sim2real

Introduction

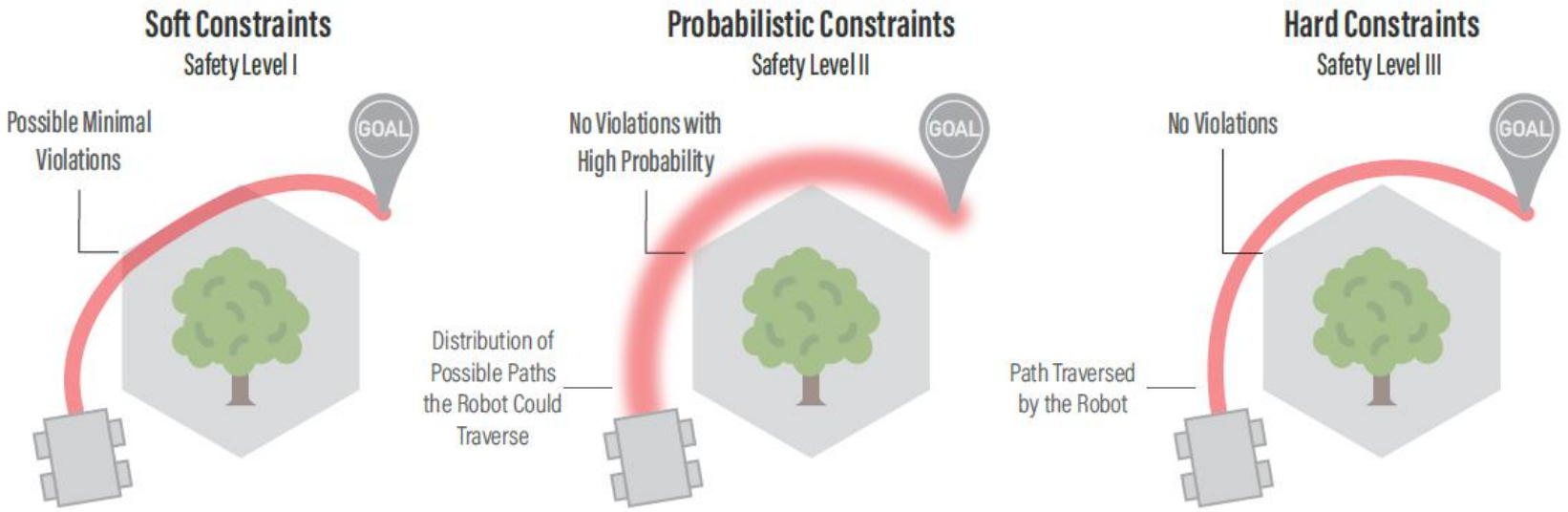
- We can't train a real robot from the scratch! Failure is not allowed
  - A robot is very expensive and delicate
  - For autonomous driving, we should not collide with the obstacles



- Safety is essentially defined by constraints in an MDP
  - We can inject constraints on the policy ($\pi \in \Pi$) or the state ($x_t \in X$)

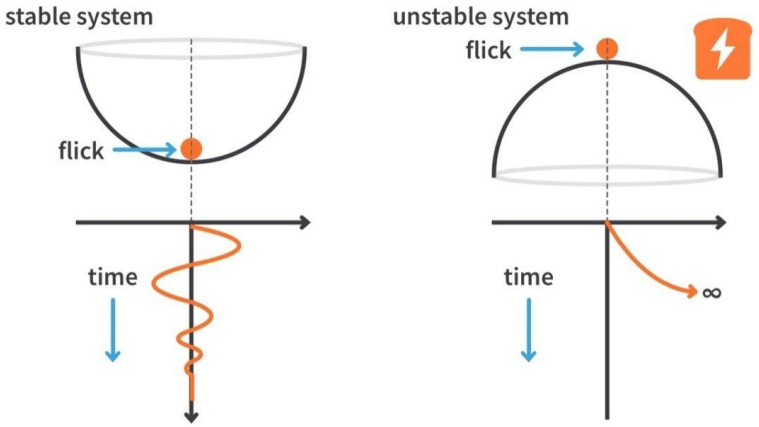
Safety

- The notion of safety is deeply connected to control theory
- Three different levels for constraint satisfaction
  - level 1: encourage (penalty to objective term)
  - level 2: probabilistic guarantees ($p(c(x_k, u_k, w_k) \leq 0) \geq p_{th}$)
  - level 3: hard constraints ($c(x_k, u_k, w_k) \leq 0$)



Safety

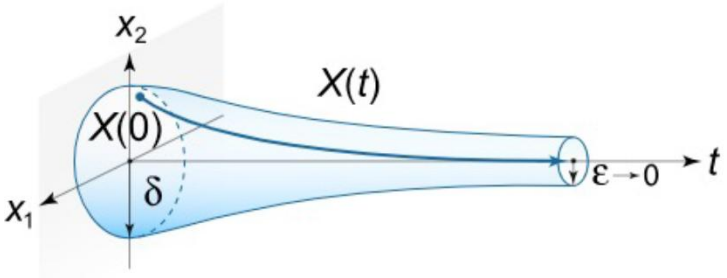
- Stability: system is stable if it is not diverging
  - Lyapunov stability: if the system is close to an equilibrium point, it remains within a small neighborhood $\forall b, \|b - a\| < \delta \rightarrow \|x(t,b) - x(t,a)\| < \epsilon$
  - Asymptotic stability: trajectories converge to the equilibrium as time goes to infinity $\forall b, \|b - a\| < \delta \rightarrow \lim_{t \to \infty} \|x(t,b) - x(t,a)\| = 0$
  - Exponential stability: the convergence to the equilibrium occurs at an exponential rate $\|x(t,b) - x(t,a)\| < \alpha \|b - a\| e^{-\beta t}, \forall t$



Safety

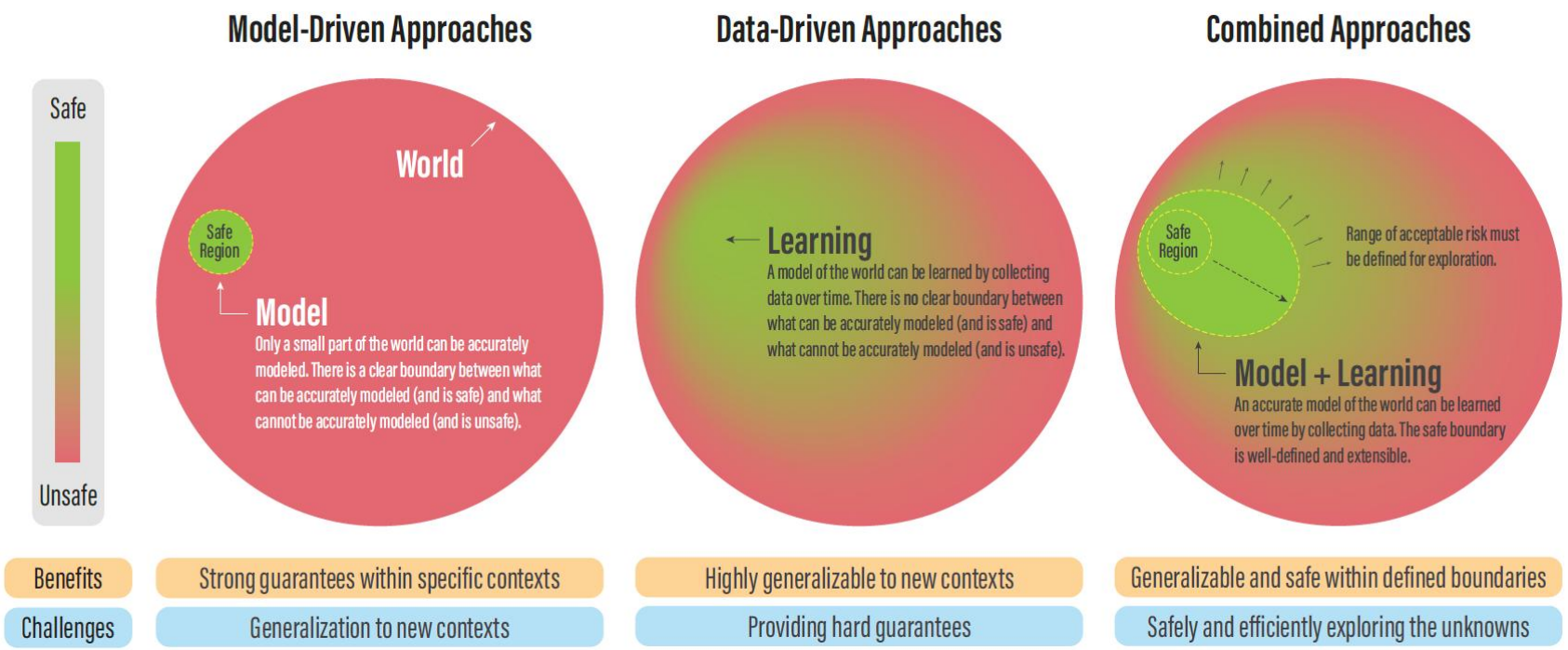
- Linear system stability
  - linear system: $x_{t+1} = Ax_t + Bu_t$
  - linear policy: $u_t = -Kx_t$
  - closed-loop system: $x_{t+1} = (A - BK)x_t$

  - stable if and only if $\rho(A - BK) \leq 1$
    - $\rho(x)$ is the spectrum radius function (maximum of the absolute value of x's eigenvalues)
    - exponentially stable if and only if $\rho(A - BK) < 1$



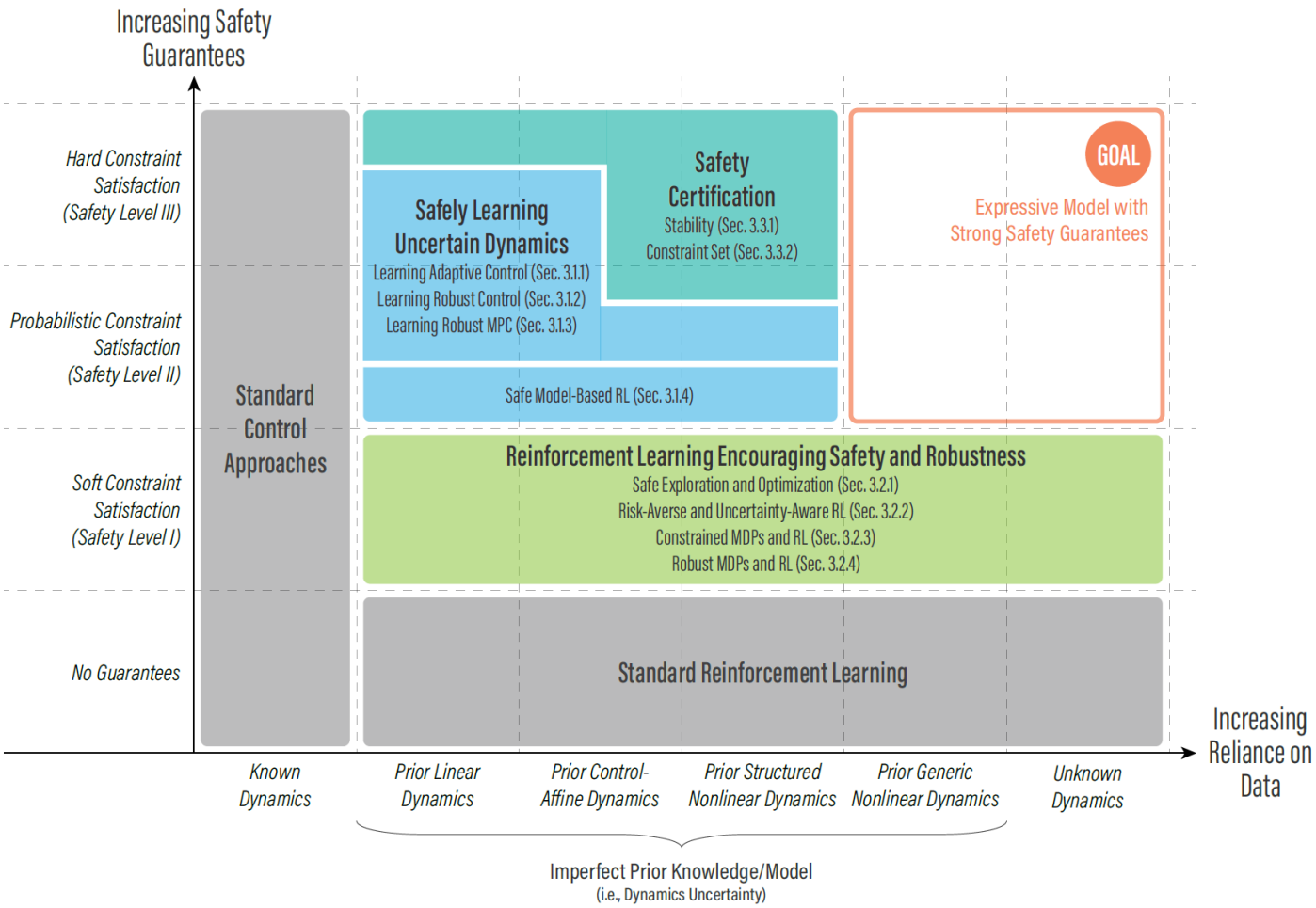
Safety

- Tradeoffs between safety and generalizability
  - theoretical strength in model-driven approaches
  - practicality in data-driven approaches (RL)



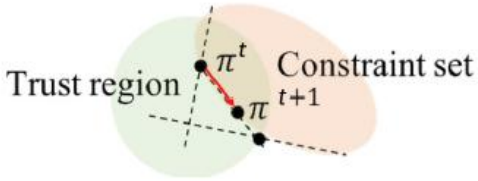
Safety

> Very diverse and active research field!



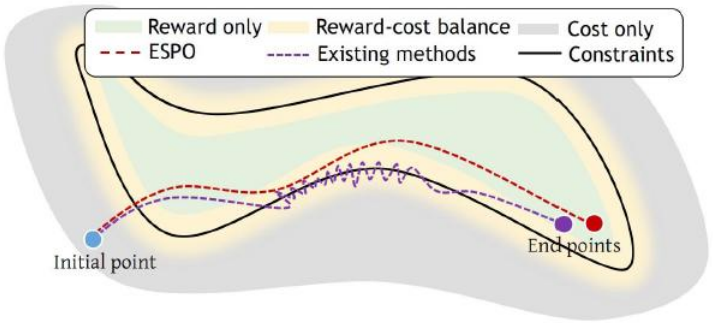
Safe model-free RL

- Constrained MDP
  - $\max_{\pi} J(\pi) = \mathbb{E}_{\pi}[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t)]$ subject to $C_i(\pi) \leq d_i$
  - Highly nonconvex, very hard to solve
  - Typically, use Lagrangian methods $\mathcal{L}(\pi, \lambda) = R(\pi) - \sum_i \lambda_i (C_i(\pi) - d_i)$
  - Algorithms
    - fix λ and perform policy gradient update (e.g. TRPO, DDPG)
    - fix π and perform dual update (e.g., PDO, CPO)
    - PDO: $\lambda_i^{k+1} = \left[\lambda_i^k + \beta(C_i(\pi_k) - d_i)\right]^+$, $[x]^+ = \max(0, x)$
    - CPO: derives the dual variable λ_i^{k+1} by solving an optimization problem



Safe model-free RL

- Primal-based methods
  - $\mathcal{L}(\pi, \lambda) = R(\pi) - \sum_i \lambda_i (C_i(\pi) - d_i)$
  - only optimize reward if no safety violation happens $\max_{\pi} R(\pi)$
  - only optimize safety if safety violation happens $\min_{\pi} \lambda_i (C_i(\pi) - d_i)$
  - Pros and cons
    - it is not sensitive to the initial points
    - do not need to find tune the dual parameters
    - it is not sample efficient
    - further investigation is required for multiple constraints
  - Algorithm examples
    - CRPO (2021)
    - PCRPO (2024)
    - ESPO (2024)



Safe model-based RL

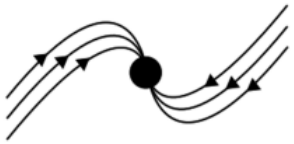
- An approach combining model learning, RL, and safety constraints
  - Model learning: learn dynamics model with uncertainty
    - Gaussian processes, Ensemble networks
  - Safety constraints: constraints formulation and enforcement
    - control barrier functions, Lyapunov stability, reachable sets, STLs
  - Policy optimization: improve policy within safety limits
    - constrained policy gradient, safe Q-learning, robust MPC
  - Safe exploration: balance safety and information gain
    - cautious exploration, guided exploration

Safe model-based RL

> Control theoretic certificates

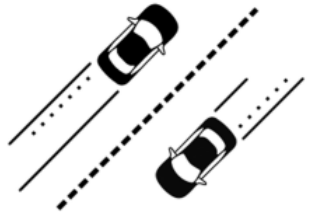
Lyapunov Function

Certifies stability of a fixed point



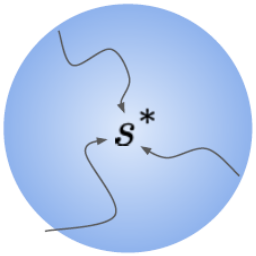
Barrier Function

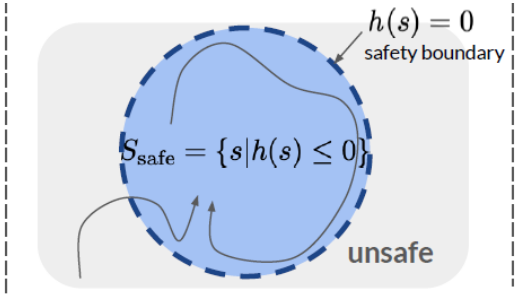
Certifies invariance of a region

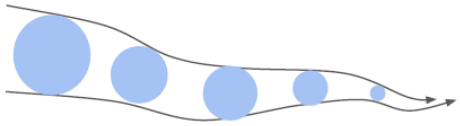


Contraction Metric

Certifies ability to track arbitrary trajectories

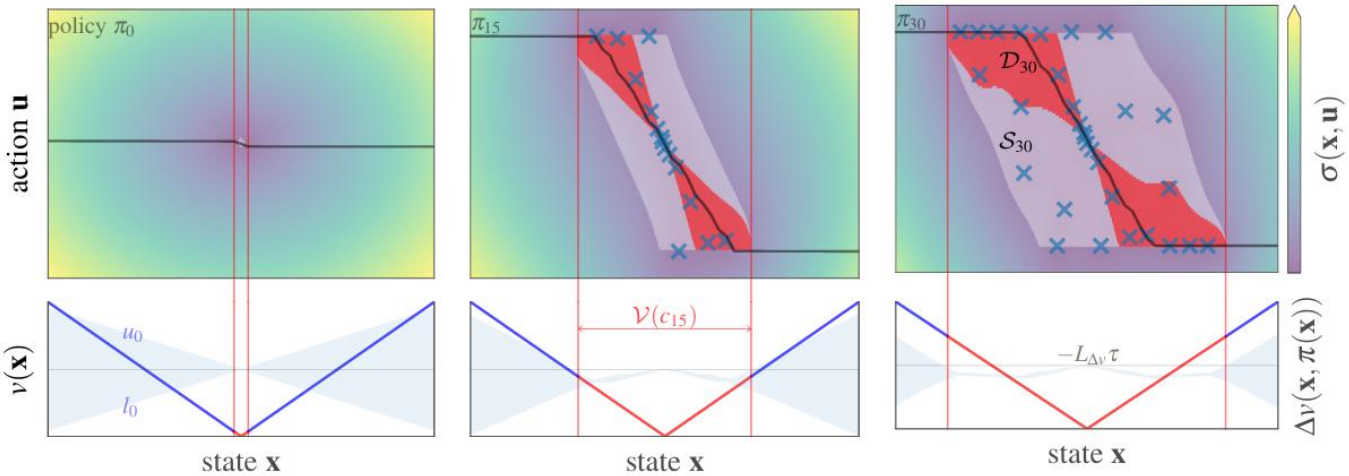






Safe model-based RL

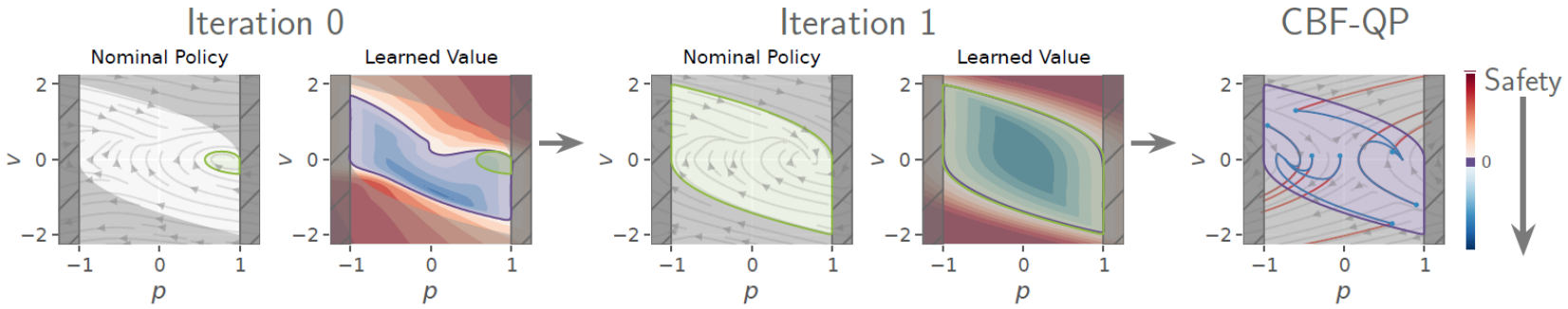
- Lyapunov function $V(x)$ is a pseudo-energy function
  - strictly decreasing along trajectories of the system
  - can provide a sufficient condition for stability
  - Key ideas of Lyapunov learning
    - learn or design a Lyapunov certificate
    - learn a policy to maximize the stability region
  - Ex. learn an inverted pendulum policy without ever falling down



(a) Initial safe set (in red). (b) Exploration: 15 data points. (c) Final policy after 30 evaluations.

Safe model-based RL

- Control barrier function ensures the system stay within safe region
  - defines a safe set $S = \{x: h(x) \geq 0\}$
  - For control-affine systems $\dot{x} = f(x) + g(x)u$, enforce $\dot{h}(x,u) + \alpha\big(h(x)\big) \geq 0$
  - can provide a sufficient condition for forward invariance
    - if the initial state is safe, then the whole trajectory will be safe
  - CBF-QP: find the closest safe action to the nominal action (e.g., RL policy)
    - act as a safety filter which projects unsafe action to be safe



Safe model-based RL

- Other methods
  - Hamilton-Jacobian reachability
  - Predictive methods
  - refer “data-driven safety filters”

Hamilton-Jacobi Reachability	
Pros	Cons
• Maximal Safe Set • Safety Certification of Systems	• Curse of Dimensionality • Indirect Synthesis of Filter

HJ↔PSF: Optimal Control Based

HJ→PSF: Maximal Safe Set [16]
HJ←PSF: Predictive Filtering

CBF↔HJ: Explicit Safe Sets

CBF→HJ: Smooth Filtering [15]
CBF←HJ: Maximal Safe Set [91]

Predictive Control	
Pros	Cons
• Scalable to Large-Scale Systems • Predictive Decision Making	• Complexity of Robust Design • Heavy Online Computation

Control Barrier Functions	
Pros	Cons
• Ease of Implementation • Smooth Safety Filtering	• CBF Synthesis • Instantaneous Decision Making

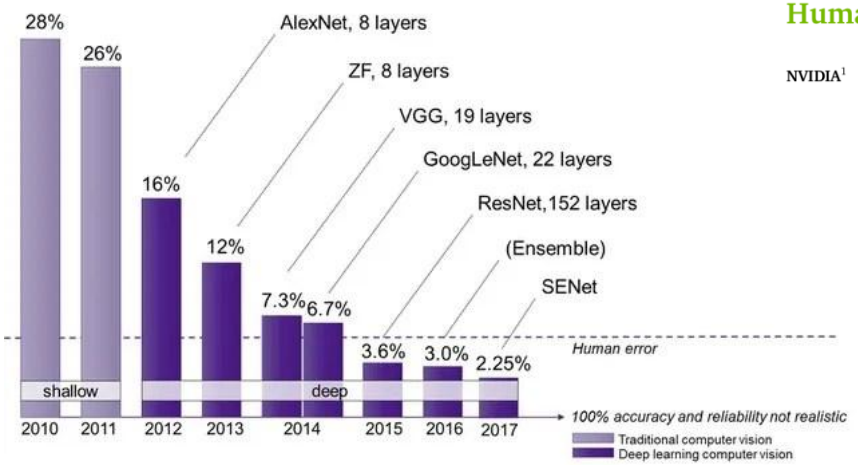
PSF↔CBF: Smooth Filtering

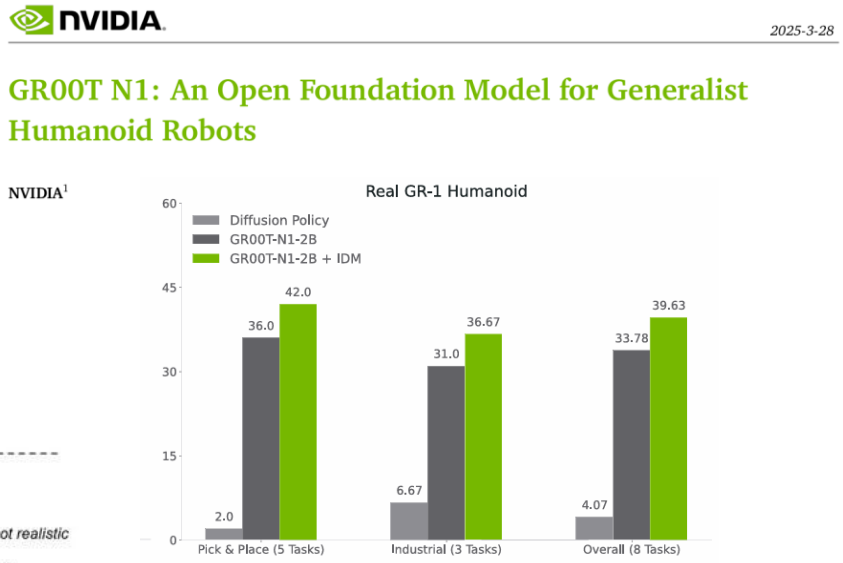
PSF→CBF: Enlarged Safe Set [46]
PSF←CBF: Robustness [18]

Safety

- In general, safety cannot be perfectly guaranteed
  - Theoretical guarantees are only valid for simple models
  - Deep models remain beyond rigorous analysis
    - Distribution shift
  - yet, AI is merely an empirically effective algorithm with no certainty

- We use AI because



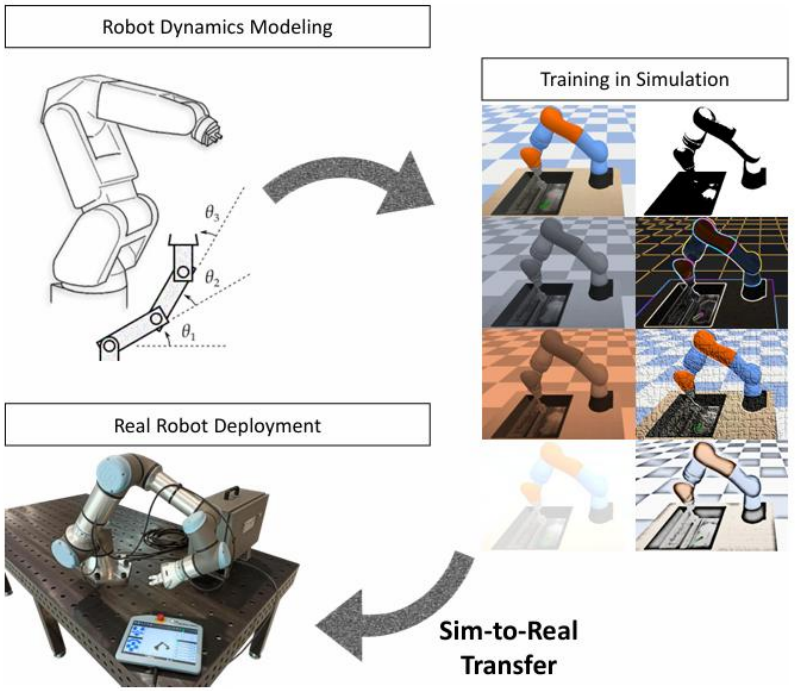


Safety

- Safety in training / test
  - do not explore the dangerous regions
    - the agent may not know where the dangerous regions are
    - RL is basically a trial-and-error algorithm, which requires experiencing failures
  - it is acceptable to encounter accidents in simulation
    - but never in test scenarios (safety should be guaranteed in test)

Sim2real

- If simulation is identical to a real world
  - train in simulation and deploy in a real world
  - digital twin (highly accurate simulators with real-world linkage)



Sim2real

- Why train in simulation?
  - real-world data is expensive
    - labor, time, setup (reset)
    - avoids wear and tear of the robot
  - fast; many RL methods require millions of samples
  - safe, can be fully explored
  - provides ground truth information

Sim2real

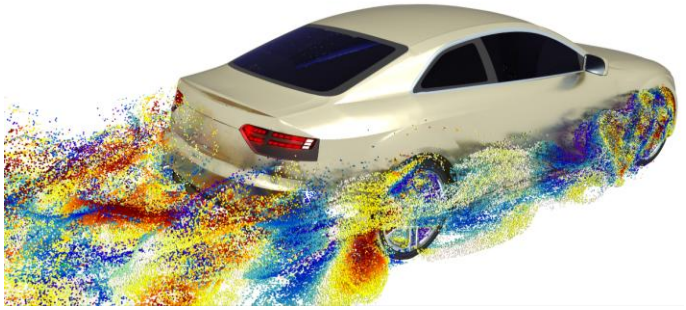
- Why not simulator?
  - dynamics challenge
    - fluid dynamics, contact dynamics, friction, deformation
    - wear, unmodeled disturbances
    - small modeling errors can accumulate
  - sensor challenge
    - vision, LiDAR, and radar often fail to reproduce unmodeled artifacts (lens flare, rolling shutter, sensor noise)
    - edge cases – mirrors, glass, water reflections, varying lights, weather
  - cannot fully capture real-world variability
    - communication delay, human factors (emotion), unpredictable behaviors





Sim2real

- Can we imitate the real world in the future?
  - simulating every particle requires as large as the physical world itself
  - approximation is inevitable
  - how many particles do we need in fluid dynamics simulators?
    - recall 1 mol = 6×10^{23} particles
    - universe cannot be a simulation

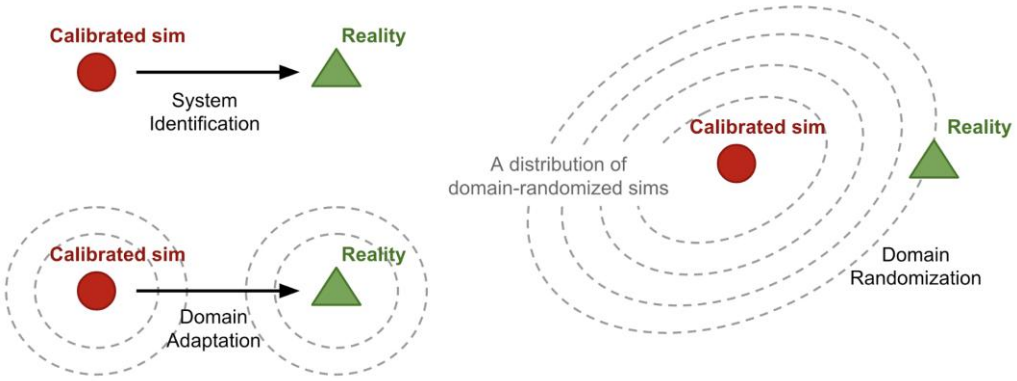


Sim2real

- Closing the reality gap
  - system identification (make simulator more accurate)

- Fine tuning
  - simulator → real-world
  - domain adaptation / transfer learning

- Robust policy learning
  - robust RL
  - invariant feature learning
  - domain randomization

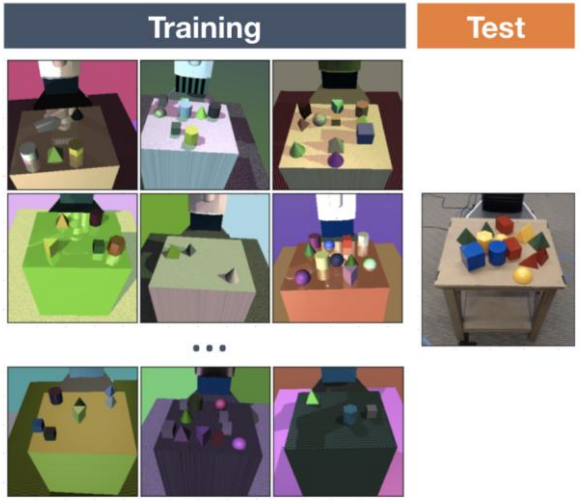


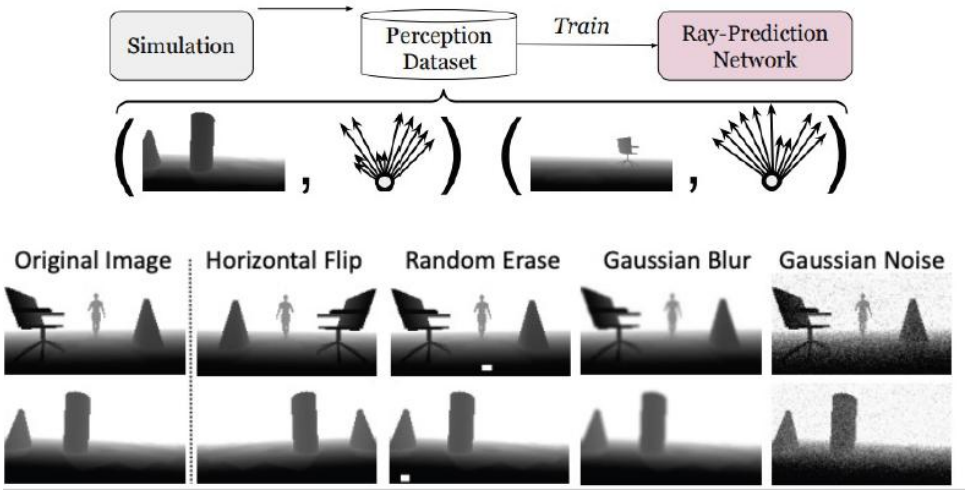
Domain randomization

- Simulator: source domain, physical world: target domain
  - training in source domain
  - we can control a set of randomization parameters in the source domain
  - The trained policy is exposed to a variety of environments and learns to generalize
  - Discrepancies between the source and target domains are modeled as variability in the source domain

Domain randomization

- Uniform domain randomization
  - each randomization parameter is uniformly sampled within the range
  - visual appearances
    - position, shape, and color objects
    - material texture
    - lighting condition
    - random noise added to images
    - position, orientation, and field of view of the camera in the simulator





Domain randomization

- Uniform domain randomization
  - physical dynamics parameters
    - mass and dimensions of objects
    - mass and dimensions of robot bodies
    - damping, friction coefficient, stiffness
    - gains for the (PID) controller
    - joint limit
    - action delay
    - observation noise (sensor noise)

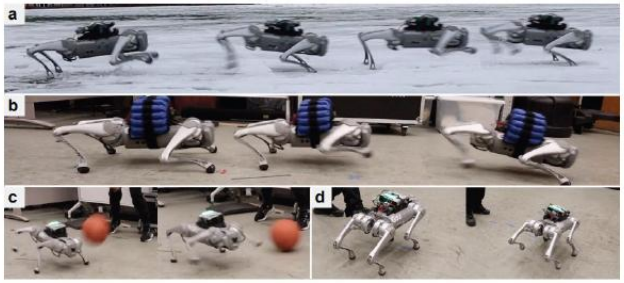
properties. The center of mass positions, the masses of links, and joint positions are randomized by adding a noise sampled from $U(-2, 2)$ cm, $U(-15, 15)$ %, and $U(-2, 2)$ cm, respectively.

	mean	standard deviation
base position	$[0, 0, 0.55]^T$	1.5 cm
base orientation	$[1, 0, 0, 0]^T$	0.06 rad (about a random axis)
joint position	$[0, 0.4, -0.8, 0, 0.4, -0.8, 0, -0.4, 0.8, 0, -0.4, 0.8]^T$	0.25 rad
base linear velocity	$\mathbf{0}^3$	0.012 m/s
base angular velocity	$\mathbf{0}^3$	0.4 rad/s
joint velocity	$\mathbf{0}^{12}$	2 rad/s

Domain randomization

- Uniform domain randomization
  - physical dynamics parameters

Term	Value
Observation	
Illusion	Enabled
Joint position noise	$\mathcal{U}(-0.01, 0.01)$ rad
Joint velocity noise	$\mathcal{U}(-1.5, 1.5)$ rad/s
Angular velocity noise	$\mathcal{U}(-0.2, 0.2)$ rad/s
Projected gravity noise	$\mathcal{U}(-0.05, 0.05)$
log(ray distance) noise	$\mathcal{U}(-0.2, 0.2)$
Dynamics	
ERFI-50 [8]	0.78 N m× difficulty level
Friction factor	$\mathcal{U}(0.4, 1.1)$
Added base mass	$\mathcal{U}(-1.5, 1.5)$ kg
Joint position biases	$\mathcal{U}(-0.08, 0.08)$ rad
Episode	
Episode length	$\mathcal{U}(7.0, 9.0)$ s
Initial robot position	$x = 0, y = 0$
Initial robot yaw	$\mathcal{U}(-\pi, \pi)$ rad
Initial robot twist	$\mathcal{U}(-0.5, 0.5)$ m/s or rad/s
Goal Position	$x_{\text{goal}} \sim \mathcal{U}(1.5, 7.5)$ m $y_{\text{goal}} \sim \mathcal{U}(-2.0, 2.0)$ m
Goal Heading	$\arctan 2(y_{\text{goal}}, x_{\text{goal}}) + \mathcal{U}(-0.3, 0.3)$ rad



Term	Value
Dynamics Randomization	
Friction	$\mathcal{U}(0.2, 1.1)$
Base CoM offset	$\mathcal{U}(-0.1, 0.1)$m
Link mass	$\mathcal{U}(0.7, 1.3) \times$ default kg
P Gain	$\mathcal{U}(0.75, 1.25) \times$ default
D Gain	$\mathcal{U}(0.75, 1.25) \times$ default
Torque RFI [56]	$0.1 \times$ torque limit N · m
Control delay	$\mathcal{U}(20, 60)$ms
External Perturbation	
Push robot	interval $= 5s$, $v_{xy} = 0.5$m/s
Randomized Terrain	
Terrain type	flat, rough, low obstacles [57]

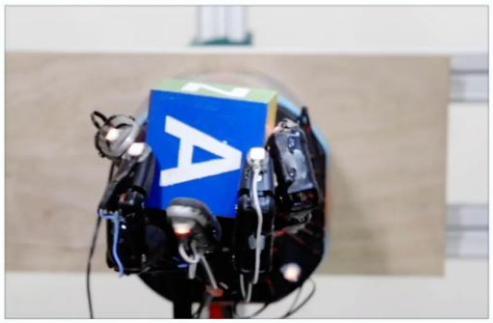
Fig. 7: Robustness Tests of our H2O system under powerful kicking. The policy is able to maintain balance for both stable and dynamic teleoperated motions.

Domain randomization

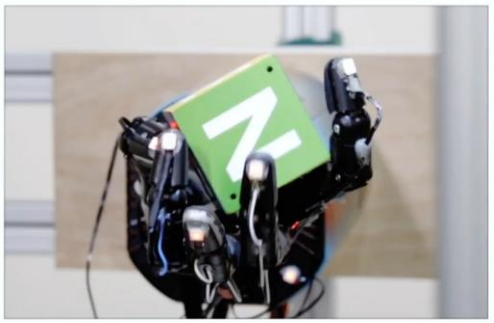
- Learning dexterous in-hand manipulation (OpenAI, 2018)
  - Randomizations developed for one object generalize to others with similar properties

Table 1: Ranges of physics parameter randomizations.

Parameter	Scaling factor range	Additive term range
object dimensions	uniform([0.95, 1.05])	
object and robot link masses	uniform([0.5, 1.5])	
surface friction coefficients	uniform([0.7, 1.3])	
robot joint damping coefficients	loguniform([0.3, 3.0])	
actuator force gains (P term)	loguniform([0.75, 1.5])	
joint limits		$\mathcal{N}(0, 0.15)$ rad
gravity vector (each coordinate)		$\mathcal{N}(0, 0.4)$ m/s^2



FINGER PIVOTING



SLIDING

FINGER GAITING

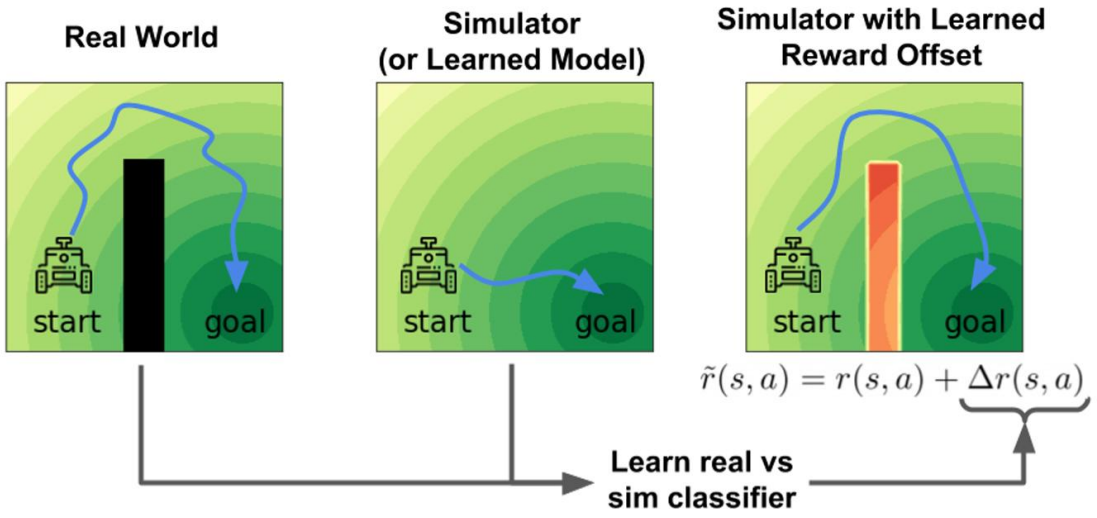
Domain randomization

- Friction

- Communication, inference delay

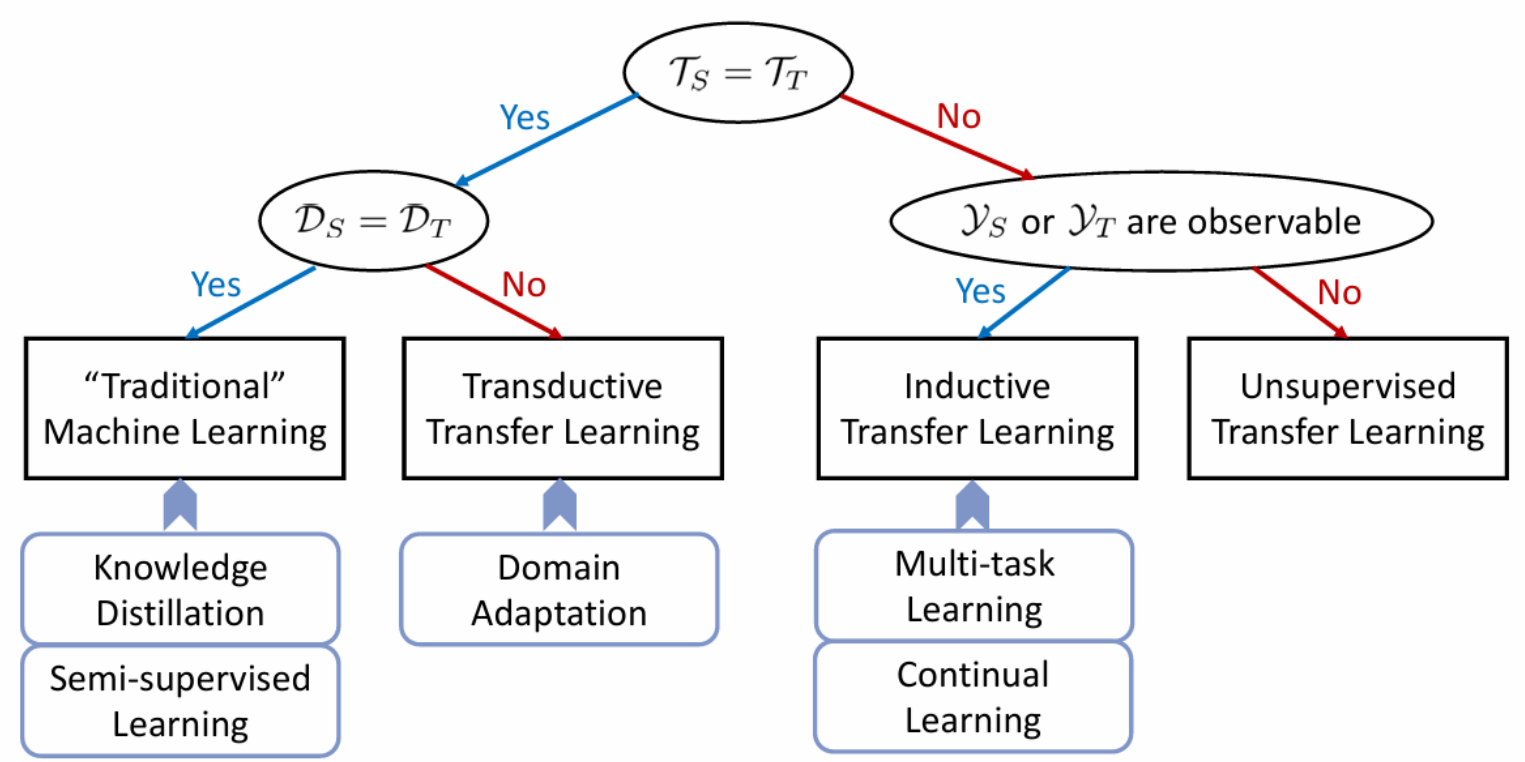
Domain adaptation

> Transfer learning: using experience from one set of tasks for faster learning and better performance on a new task
  - shot: number of attempts in the target domain
  - 0-shot: just run a policy trained in the source domain
  - 1-shot: try the task once
  - few shots: try the task a few times



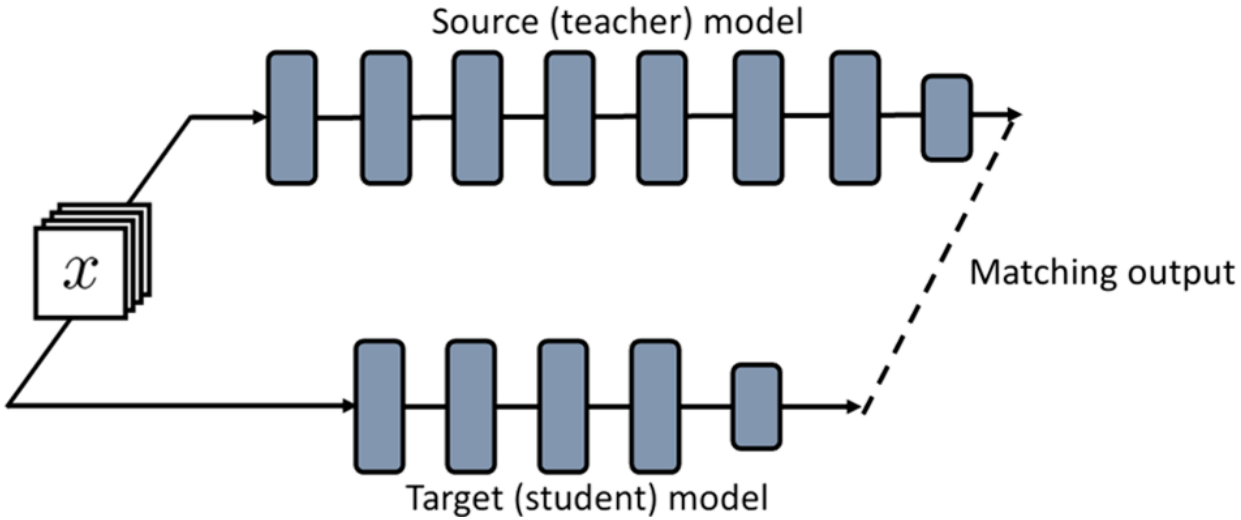
Domain adaptation

- A source domain $\mathcal{D}_S$ and learning task $\mathcal{T}_S$
- A target domain $\mathcal{D}_T$ and learning task $\mathcal{T}_T$



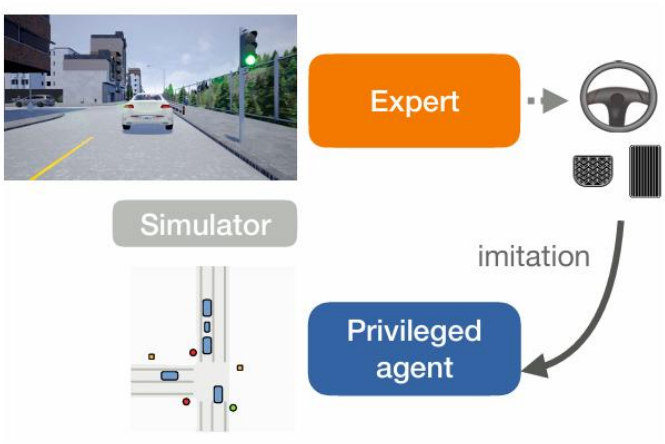
Domain adaptation

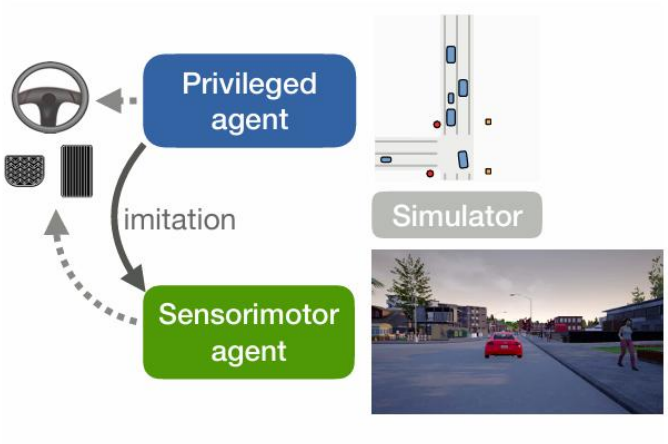
- An approximation of adaptive control
  - environment parameters are often unknown in the real
  - learning from a privileged teacher
    - 1) first train a teacher policy π_t with privilege information in simulation
    - 2) then a student policy π_s learns from π_t in simulation
    - 3) then deploy π_s in the real



Domain adaptation

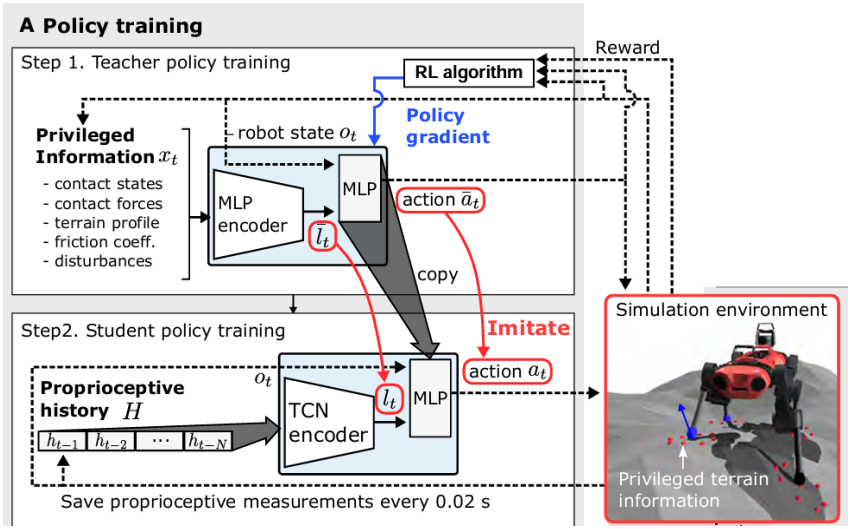
> Learning from a privileged teacher
  - it knows ground truth state (traffic light, other vehicle's pos/vel, …)
  - a sensorimotor student learns from this trained privileged agent (imitation learning)

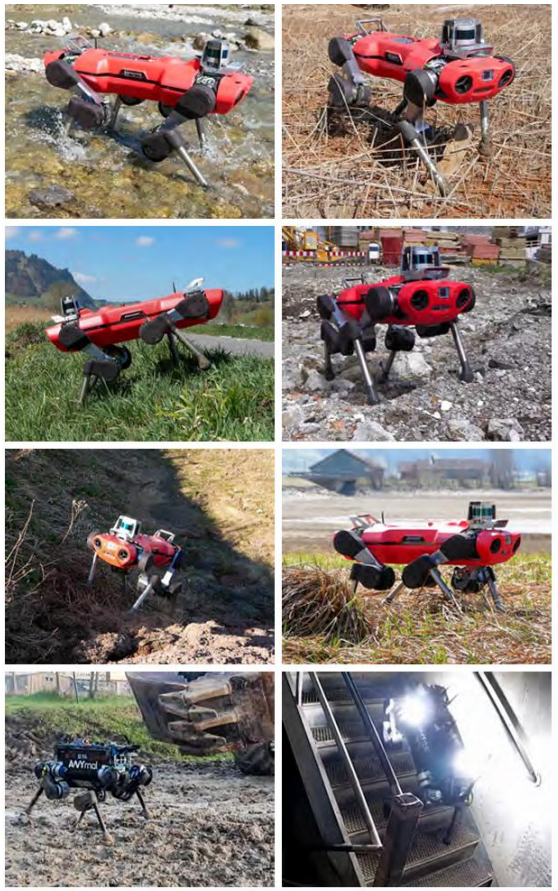


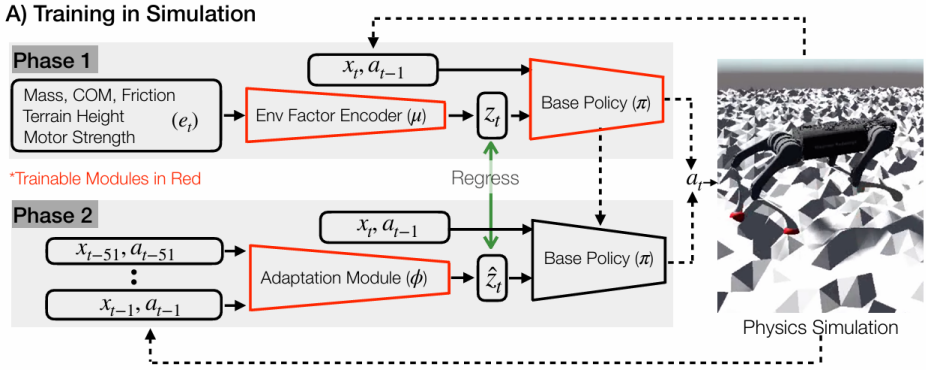


Domain adaptation

> Learning how to walk in diverse terrains

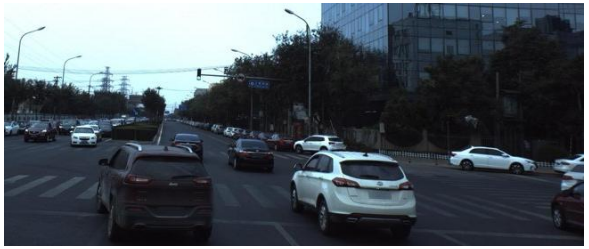






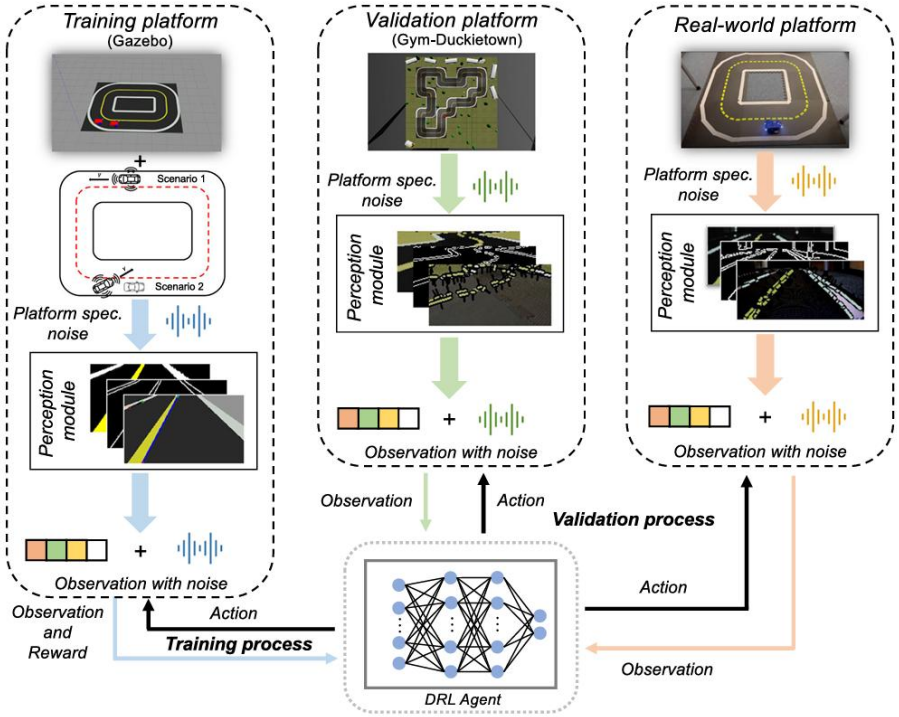
Invariant feature

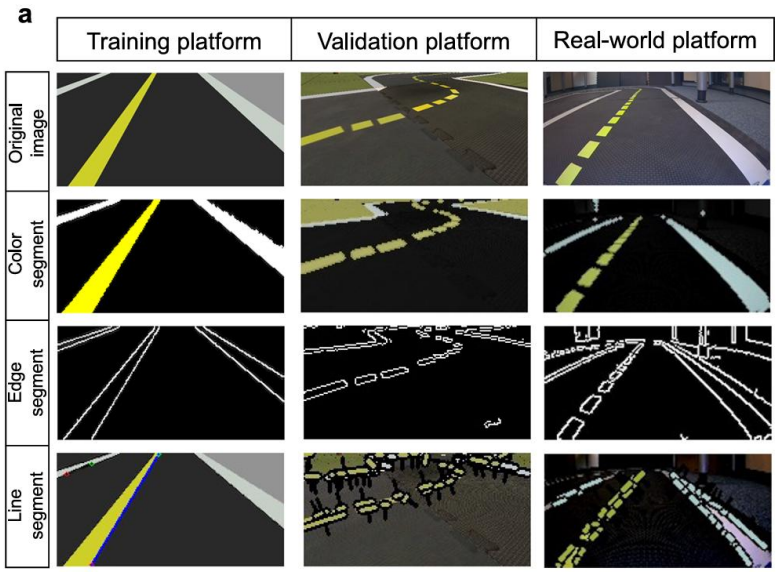
> Use domain-irrelevant input (features)

	Real-world	Simulation
Original space		
Segmentation (feature space)		

Invariant feature

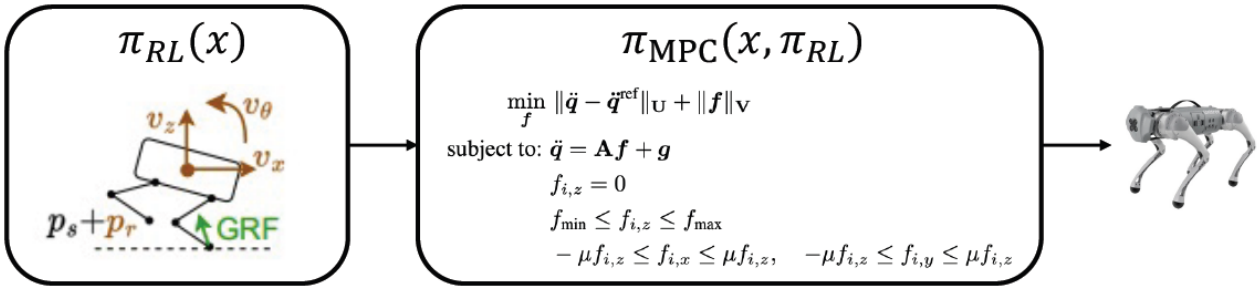
> A platform-agnostic deep RL for effective sim2real (2024)

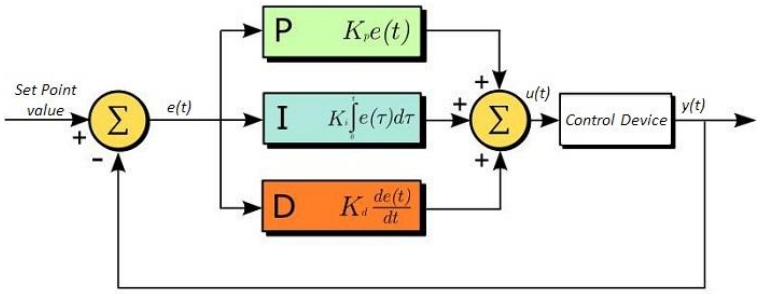




Invariant feature

- Use domain-irrelevant output (high-level policy)
  - because low-level dynamics are very different between simulation and reality
  - desired base velocity, gait frequency, swing leg residual trajectory
  - task, behavior planning



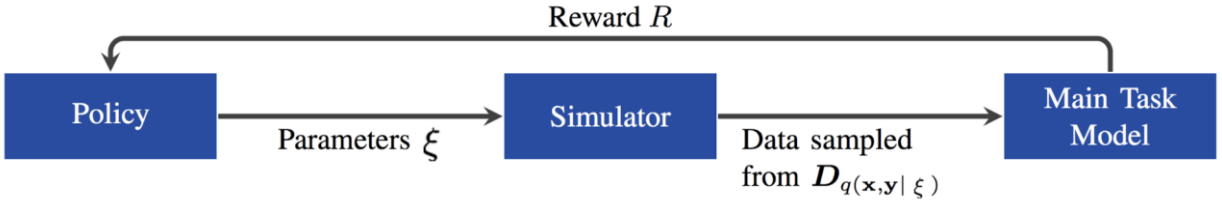


Guided domain randomization

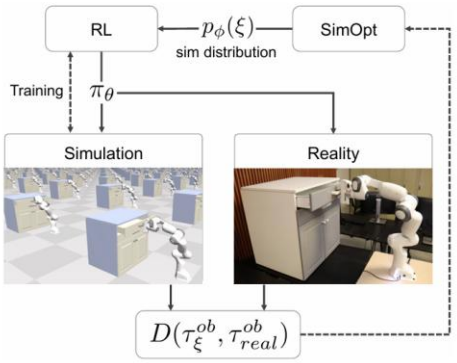
- Simulation pretraining → real-world fine-tuning
  - 1-shot or few-shot learning
  - foundation model
- If we can access to the real-world data (even if it's little), we can close the loop (sim → real → sim)
  - By measuring the discrepancies between learned behavior in simulation and in real-world, we tune the simulation's randomization parameter
  - Guided domain randomization

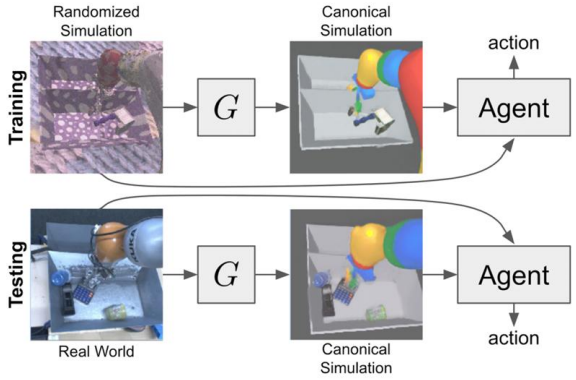
Guided domain randomization

- Optimization for task performance
  - tune randomization parameter distribution by feedback of task performance
  - Learning to simulate: task feedback as reward in RL problem



- Match real data distribution
  - learn the randomization parameter that brings the state distribution in simulator close to the state distribution in the real world

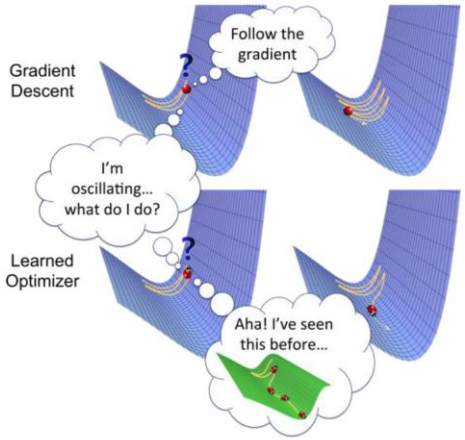




Domain randomization

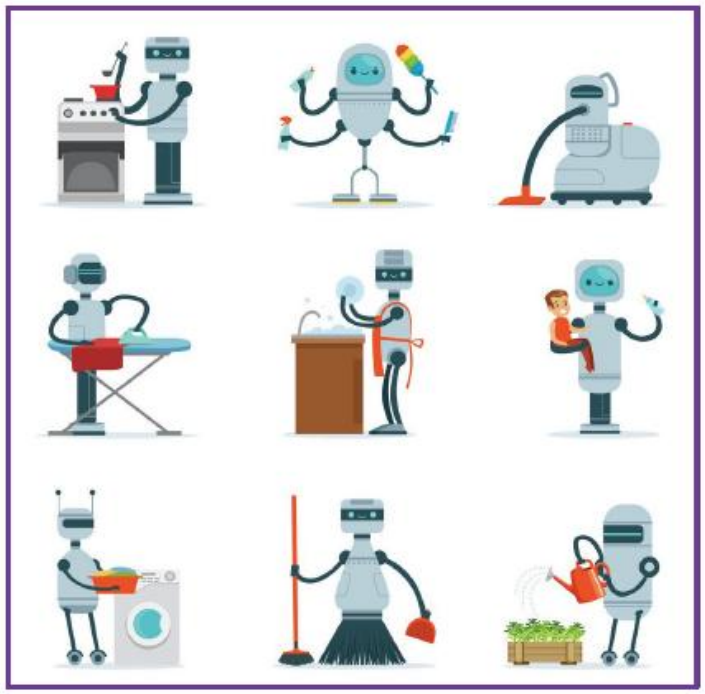
- DR as optimization
  - view learning randomization parameters in DR as a bilevel optimization
  - Mostly, a manual optimization process on tuning the randomization parameters using domain knowledge (trial-and-error experience)
  - Guided domain randomization aims to do bilevel optimization

- DR as meta-learning
  - policy to generalize across different environmental dynamics
  - domain randomization composes a collection of different tasks
  - meta-learning: learning to learn



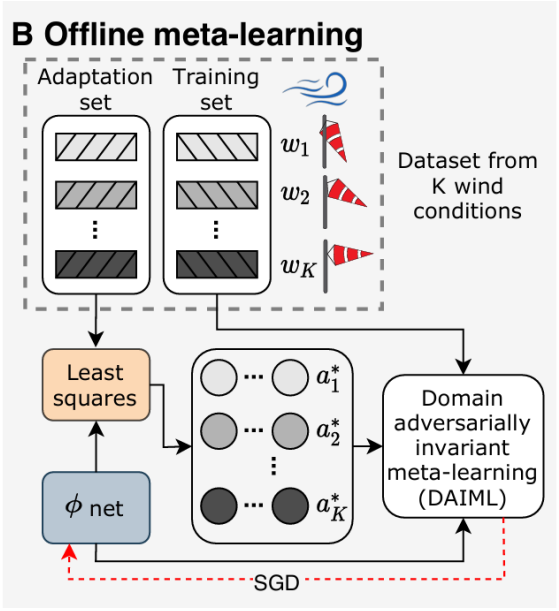
Meta-learning

- Ultimate goal of robot learning
  - build general-purpose embodied intelligence by learning to make sequential decisions in the physical world
  - humans can do thousands of tasks in thousands of environments
  - How about robots?



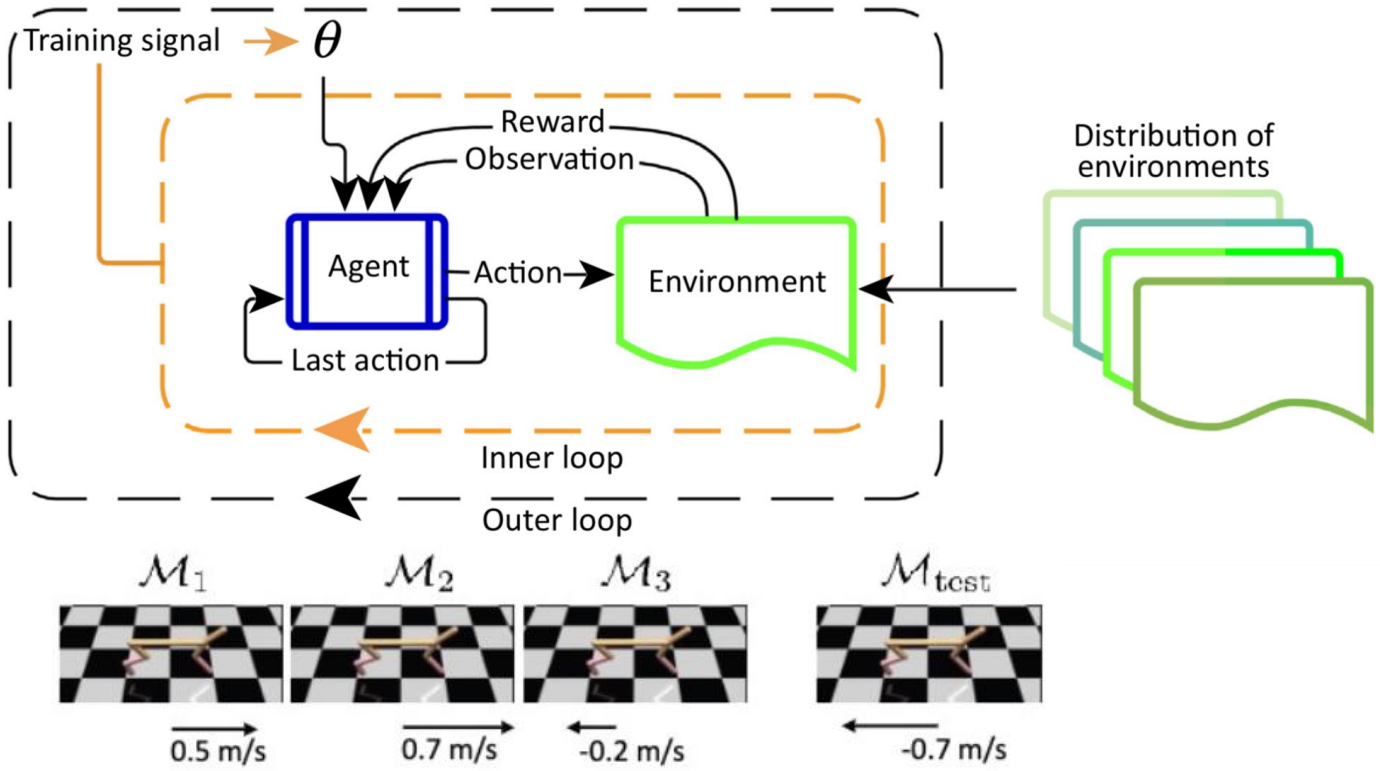
Meta-learning

- Real-world keeps changing (non-stationary dynamics)
  - hardware drift: wear and tear
  - battery and power
  - payload, center of mass shifts
  - external disturbances



Meta-learning

> A meta-RL model is trained over a distribution of MDPs, and at test time, it is able to learn to solve a new task quickly
  - can we share data across tasks?

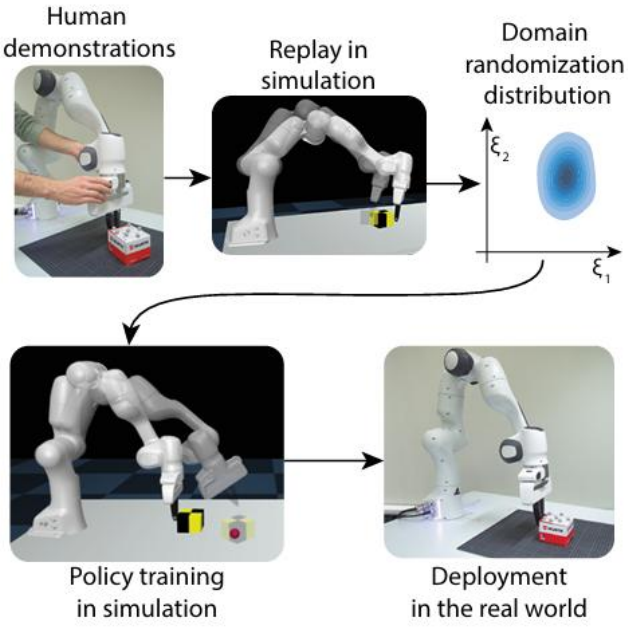


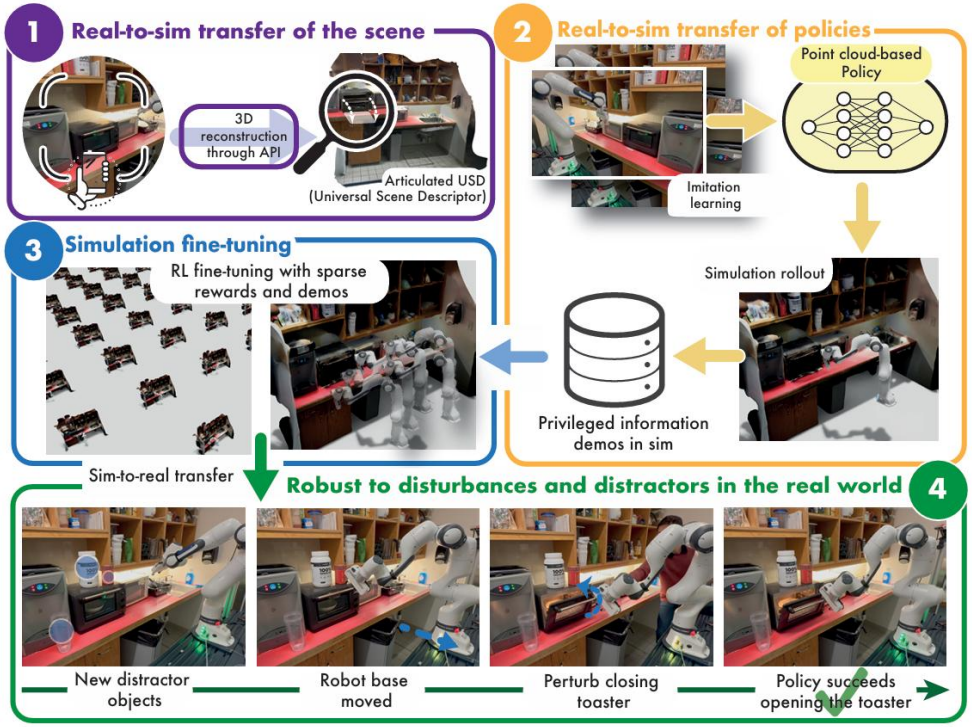
Meta-learning

- Gradient-based (MAML family)
  - Learn an initialization such that a few gradient steps on a new task yield high post-adaptation performance
- Latent / context inference
  - Infer a latent context z from recent history
  - policy is conditioned on the state and the latent context
- Recurrent meta-policy
  - feed recent history to an RNN; its hidden state internalizes the task
- Model-based meta-RL
  - apply meta-learning in dynamics model learning
  - then exploit the model (optimization / adaptive control)

Closing the loop

- Real-world: expert demonstration is possible, including env. info
- Simulation: making a policy robust (randomization)
- Real-to-sim → sim-to-real (DROPO 2022, RialTo 2024)





Reference

- Safety
  - https://arxiv.org/abs/2108.06266
  - https://arxiv.org/abs/2205.10330
  - https://arxiv.org/abs/2202.11762
  - https://ieeexplore.ieee.org/document/10266799

- Sim2real
  - https://arxiv.org/pdf/2009.13303
  - https://github.com/LongchaoDa/AwesomeSim2Real
  - https://arxiv.org/abs/2502.13187