Sim-to-Real

An Incomplete Overview

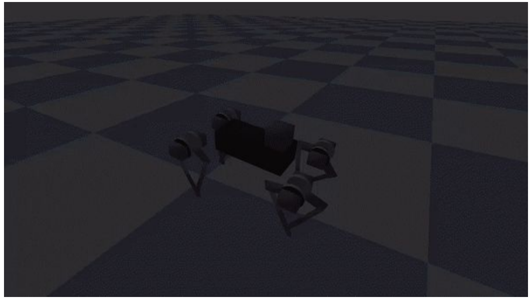
Jie Tan

CS 8803 Deep Reinforcement Learning for Intelligent Control

04/04/2022

What's the sim-to-real gap?

Dynamics:



Perception:

Goals

- Understand the causes of sim-to-real
- Review of the state-of-the-art methods
  - System Identification
  - Domain Randomization
  - Domain Adaptation
  - Meta Learning

Brainstorming: Why Sim-to-Real?

Real world

- Slow
- Unsafe
- Expensive
- Human supervision

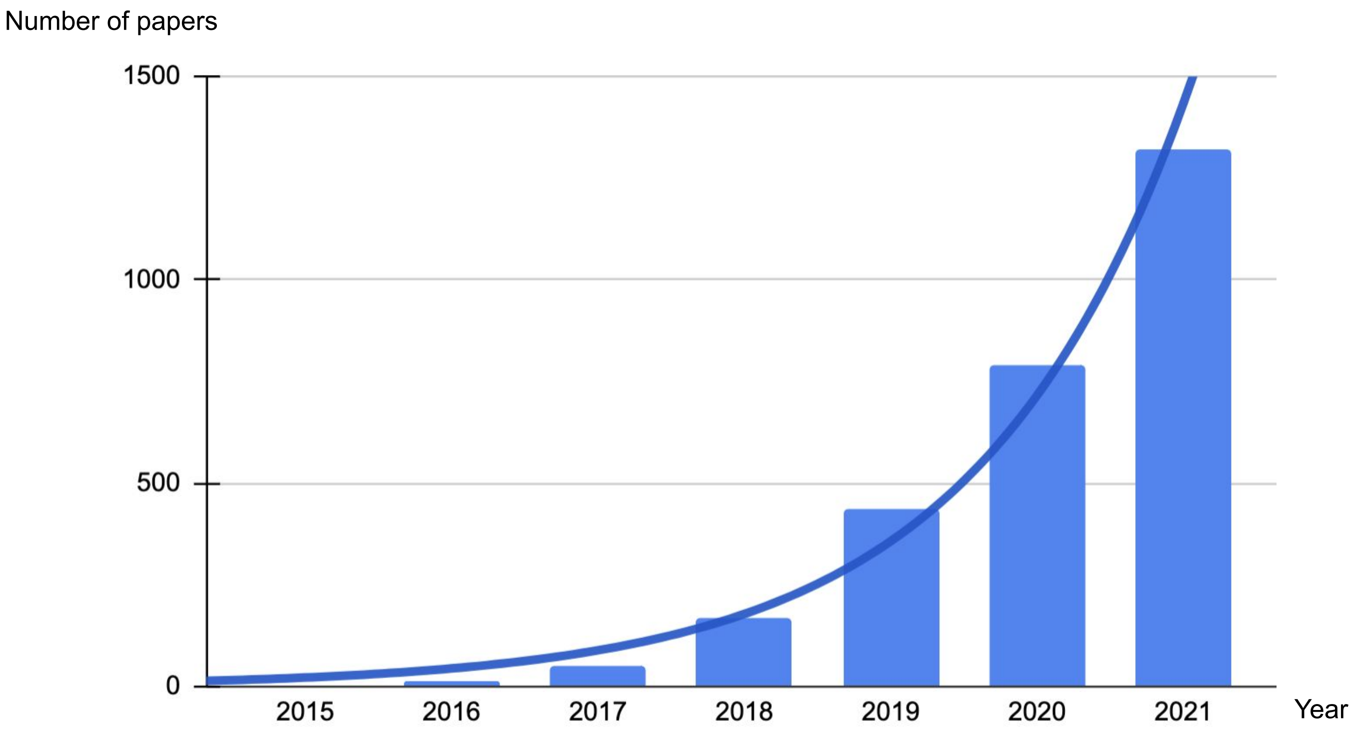
Simulation

- Fast
- Safe
- Cheap
- Scalable

Why Sim-to-Real?

Transform hard robotic problems into large-scale computation problems

Trend on Sim-to-Real



How to overcome sim-to-real gap?

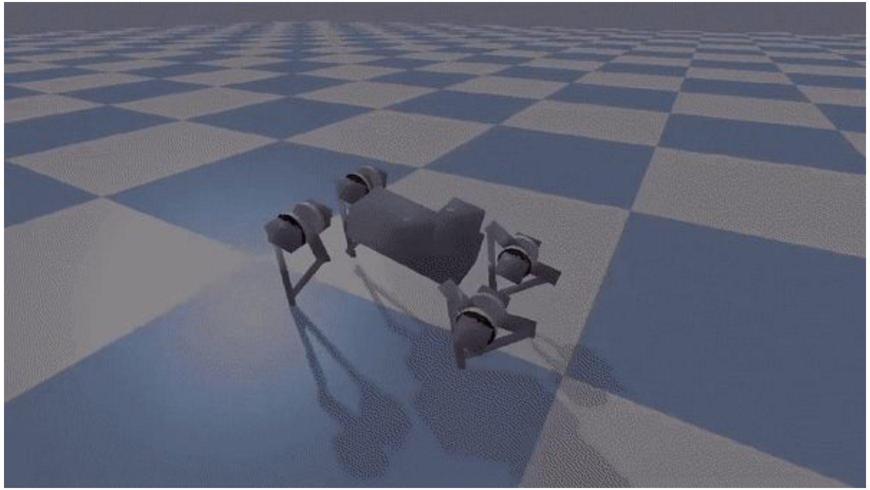
- Improve simulation
  - System identification
    - Sim-to-Real: Learning Agile Locomotion For Quadruped Robots
    - Simulation-Based Design of Dynamic Controllers for Humanoid Balancing
    - Preparing for the Unknown: Learning a Universal Policy with Online System Identification
- Improve policy
  - Domain randomization
    - Sim-to-Real Transfer of Robotic Control with Dynamics Randomization
    - Closing the Sim-to-Real Loop: Adapting Simulation Randomization with Real World Experience
  - Domain adaptation
    - Learning Agile Robotic Locomotion Skills by Imitating Animals
    - Sim-to-Real Transfer for Biped Locomotion
  - Meta learning
    - Rapidly Adaptable Legged Robots via Evolutionary Meta-Learning
    - NoRML: No Reward Meta-Learning
    - Policy Transfer with Strategy Optimization

System Identification

What are the causes of sim-to-real gap?

- Unmodeled dynamics
- Wrong simulation parameters
- Inaccurate contact models
- Latency
- Actuator dynamics
- Noise
- Stochastic real environment
- Numerical accuracy
- ...

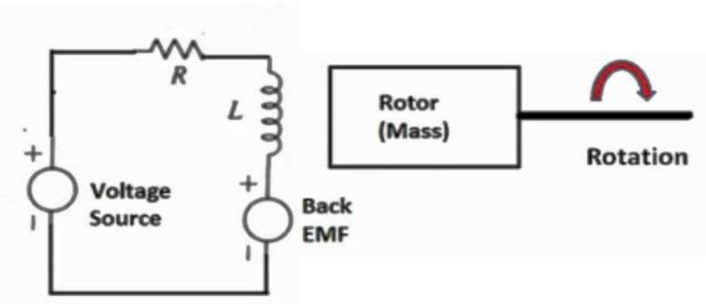
Actuator dynamics and **latency** are two important causes of reality gap.



[Sim-to-Real: Learning Agile Locomotion For Quadruped Robots, RSS 2018]

Actuator Model

Analytical models



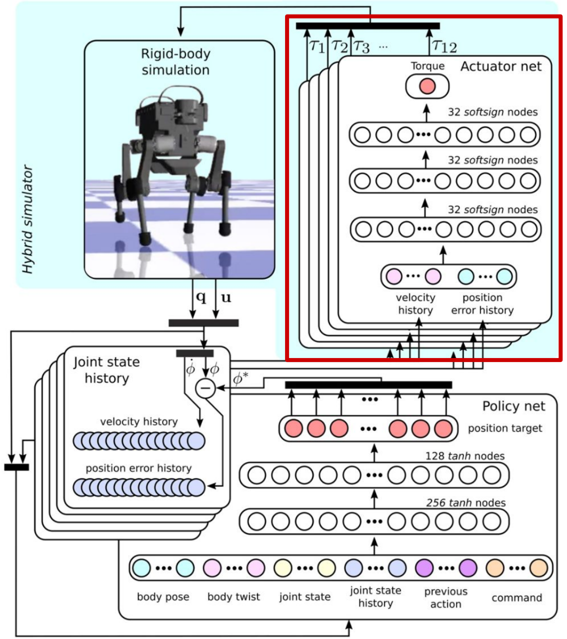
$$\tau = f(I)$$

$$I = \frac{V * \text{PWM} - V_{\text{emf}}}{R}$$

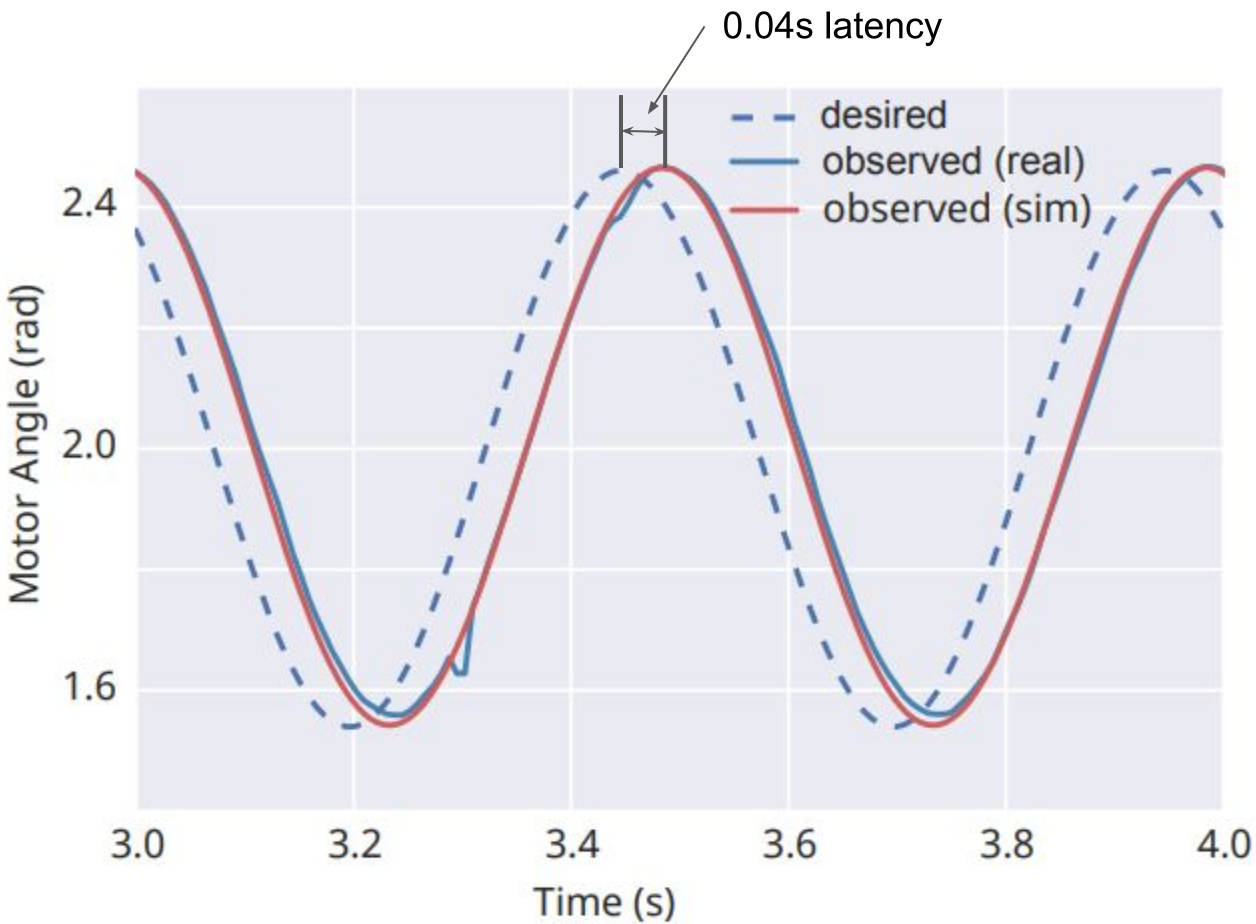
$$V_{\text{emf}} = K_t \dot{q}$$

[Sim-to-Real: Learning Agile Locomotion For Quadruped Robots, RSS 2018]

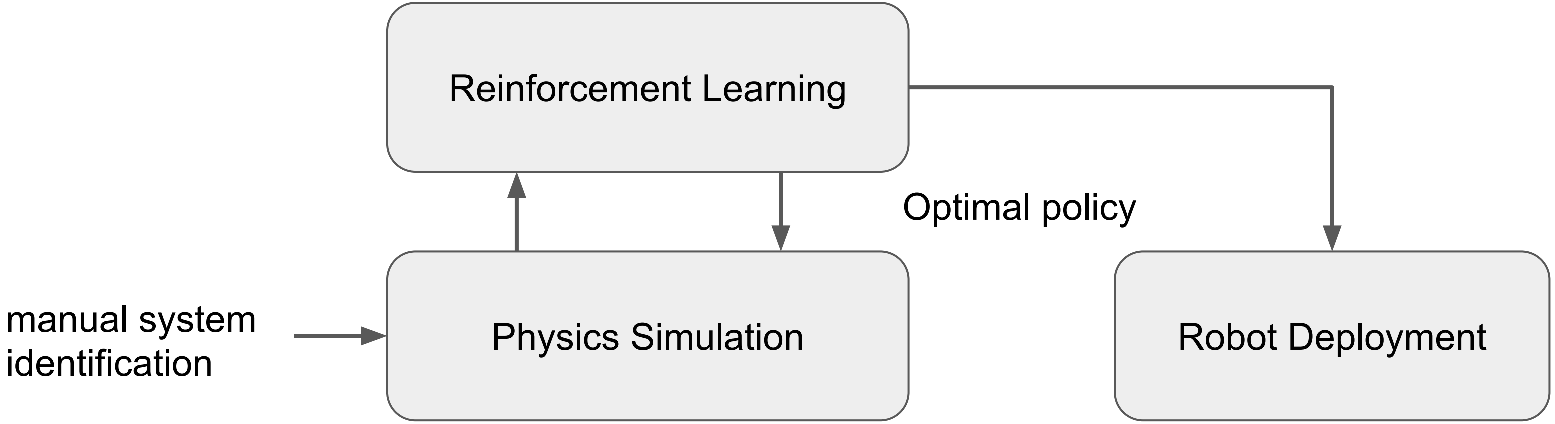
Neural network models



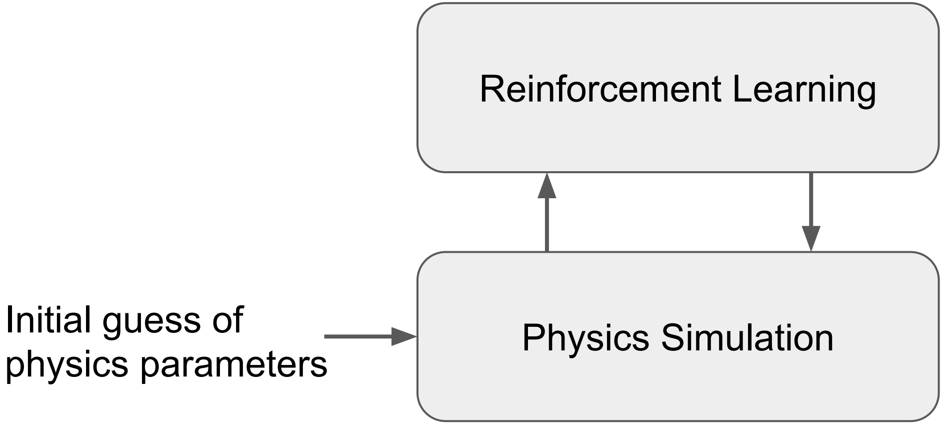
[Learning agile and dynamic motor skills for legged robots, Science Robotics 2019]



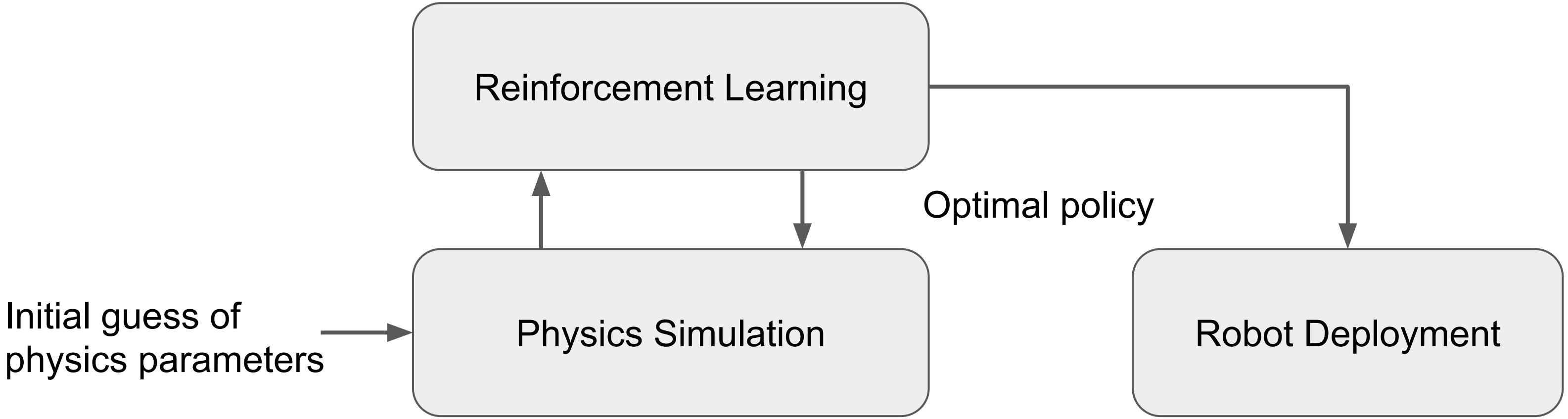
[Sim-to-Real: Learning Agile Locomotion For Quadruped Robots, RSS 2018]



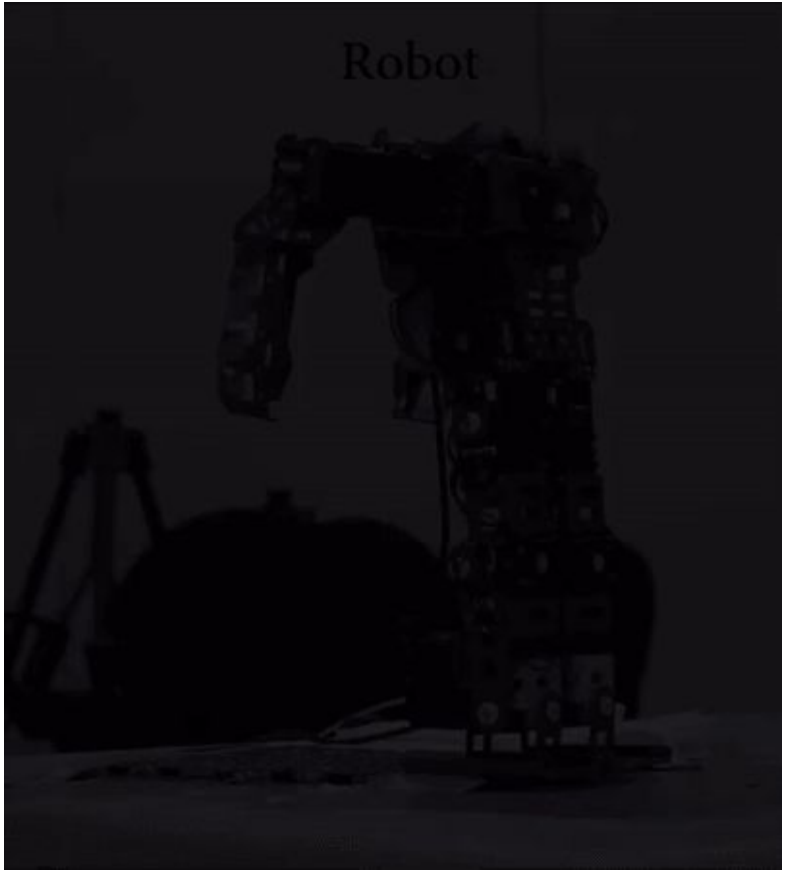
- Limitations
    - Disassemble the robot
    - Decide what parameters to identify
    - Design experiments for individual parameters
    - Lots of manual work



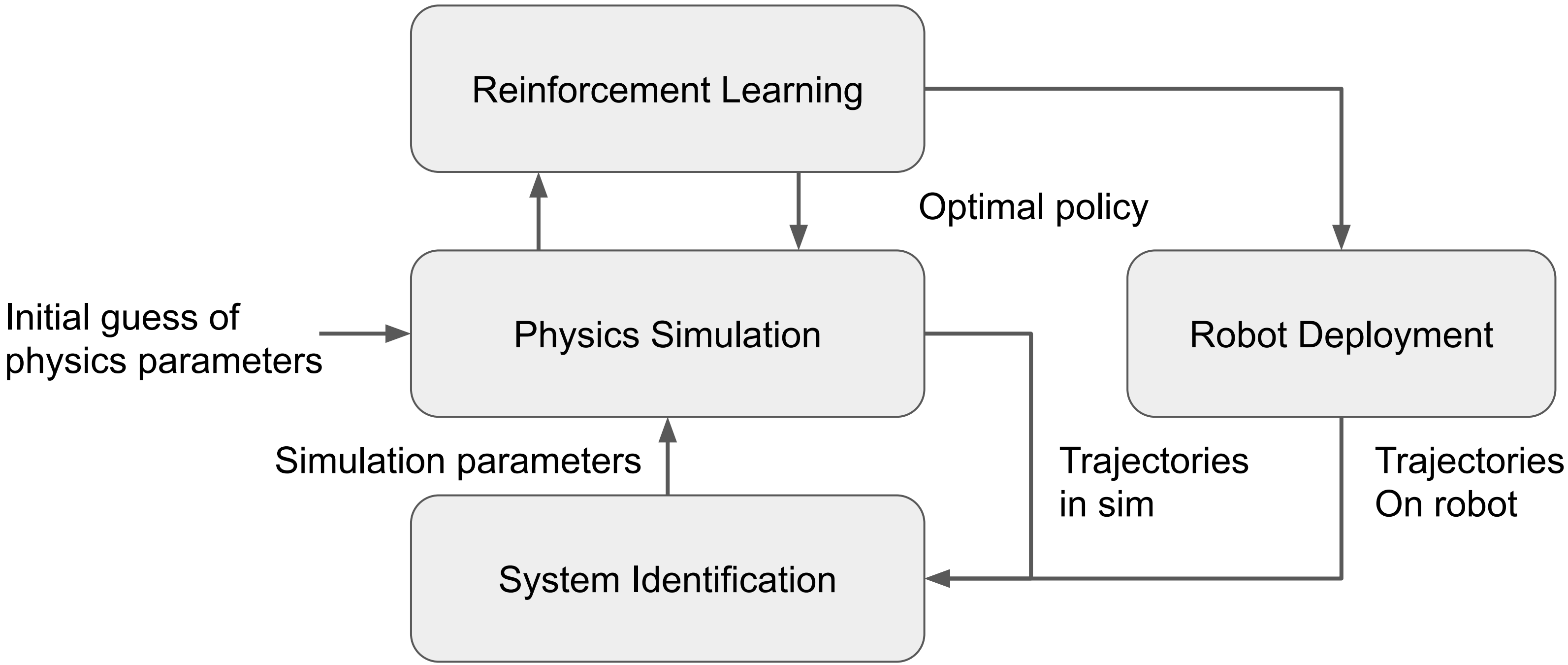
[Simulation-based design of dynamic controllers for humanoid balancing, IROS 2016]



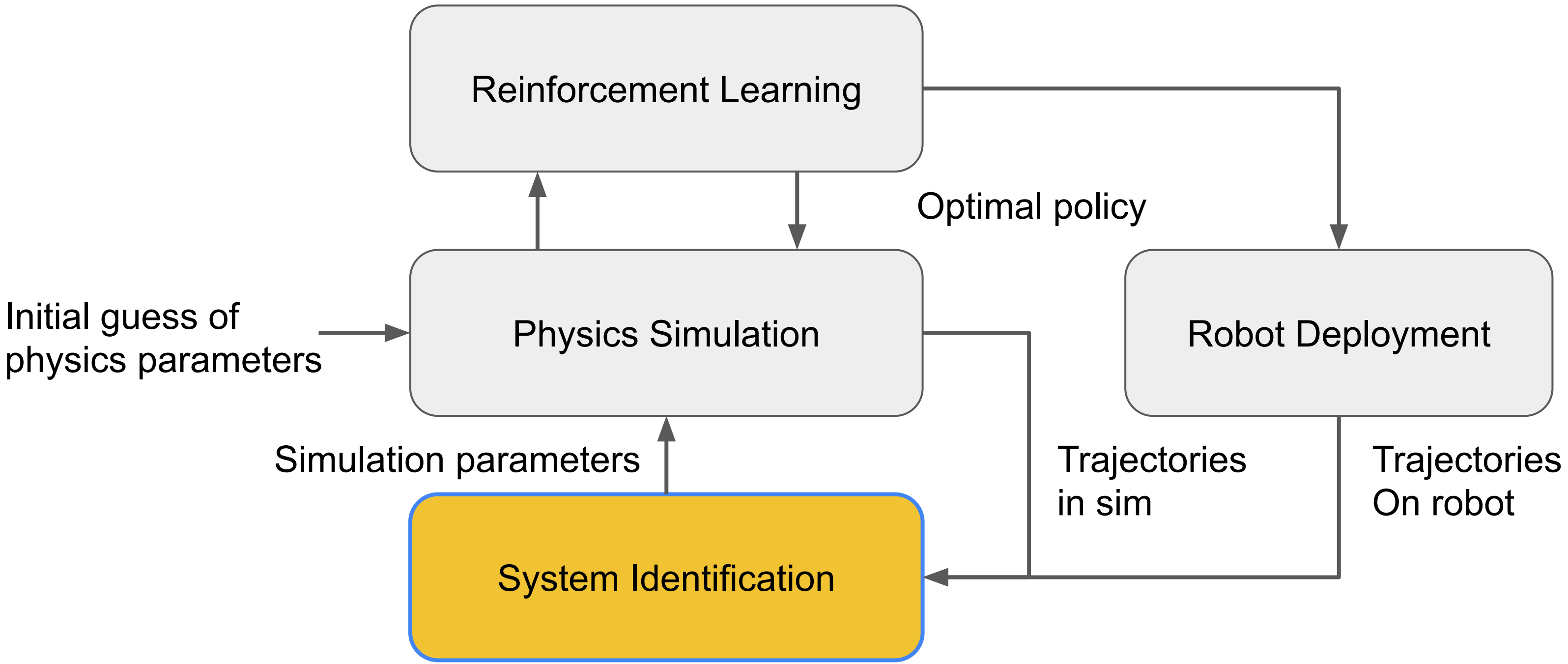
[Simulation-based design of dynamic controllers for humanoid balancing, IROS 2016]



[Simulation-based design of dynamic controllers for humanoid balancing, IROS 2016]



[Simulation-based design of dynamic controllers for humanoid balancing, IROS 2016]



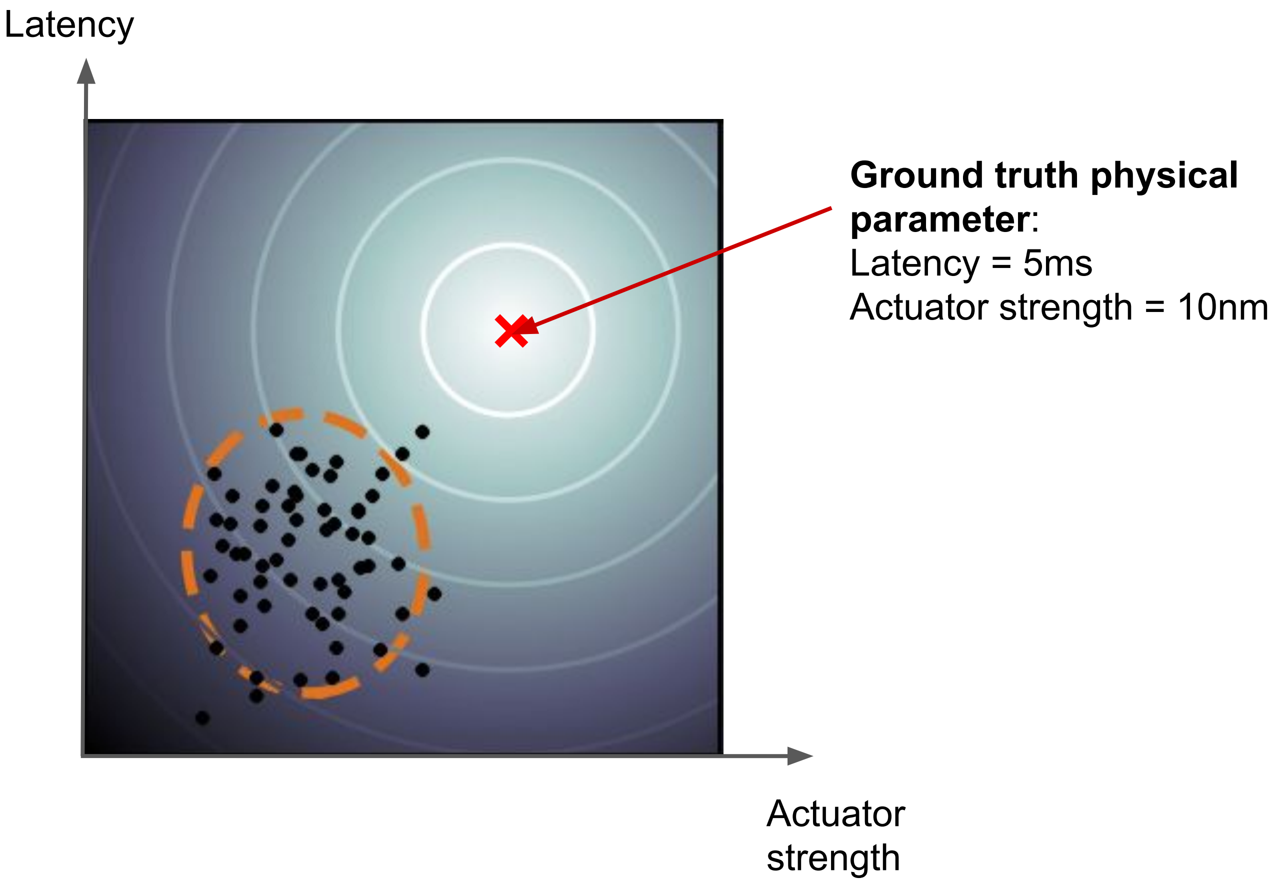
[Simulation-based design of dynamic controllers for humanoid balancing, IROS 2016]

Automatic System Identification

- Measure sim-to-real discrepancy

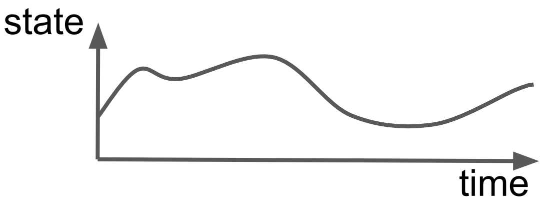
$$\boxed{\boldsymbol{\theta}} = \arg\min \boxed{\frac{1}{n}\sum_{i=1}^{n}} \int_{0}^{T+1} \|\boxed{\tilde{\mathbf{q}}_i(t)} - \boxed{\mathbf{q}_i(t;\boldsymbol{\theta})}\|_{\mathbf{W}}^{2}\,\mathrm{d}t$$

- Optimize the physics parameters
  - Covariance Matrix Adaptation-Evolution Strategy

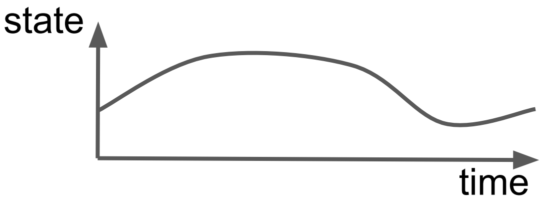

Latency
Actuator strength
Ground truth physical parameter:
Latency = 5ms
Actuator strength = 10nm

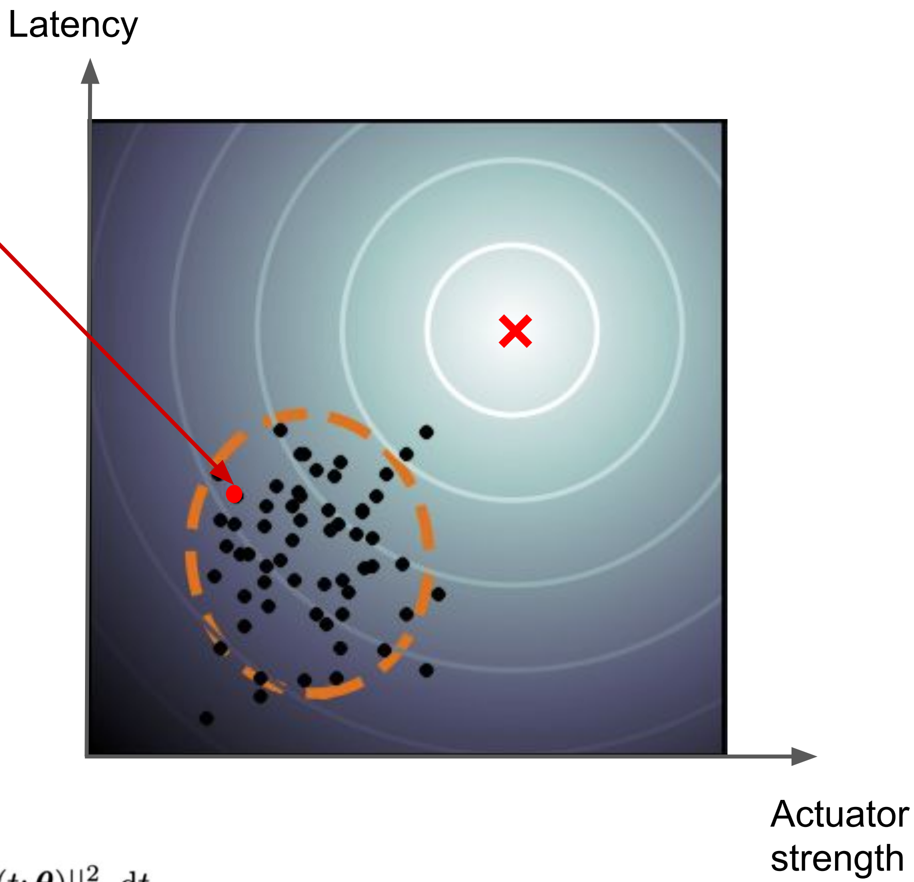
Randomly sampled physical parameter:
Latency = 1ms
Actuator strength = 2Nm

Sim trajectory:

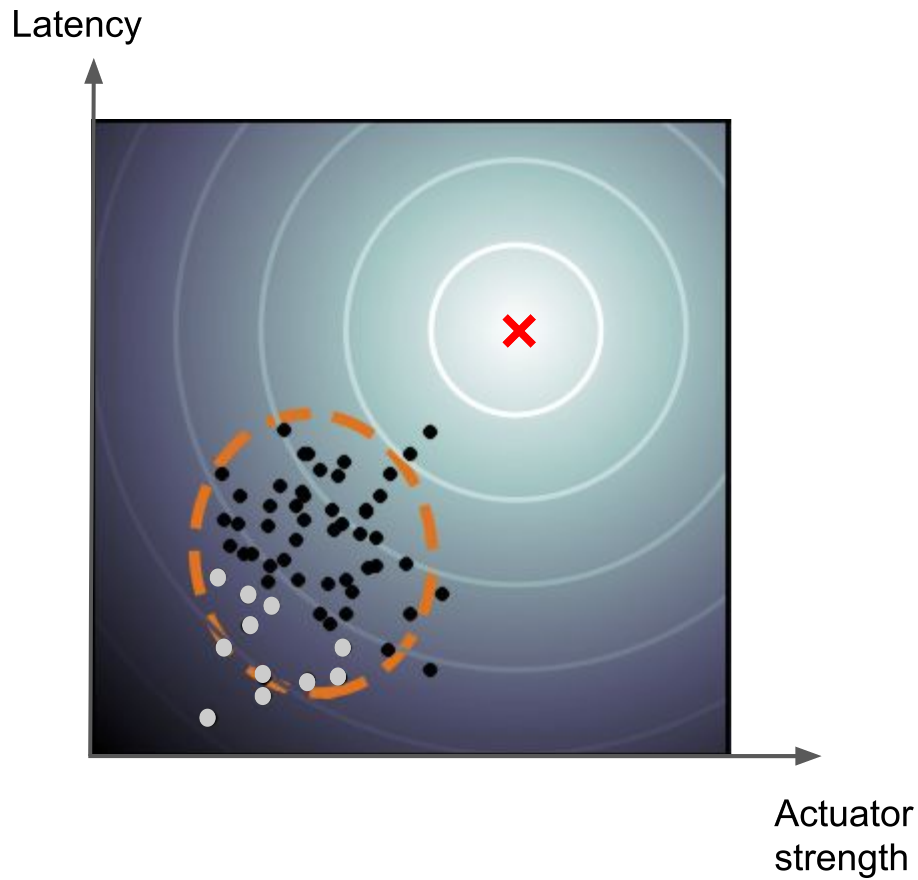


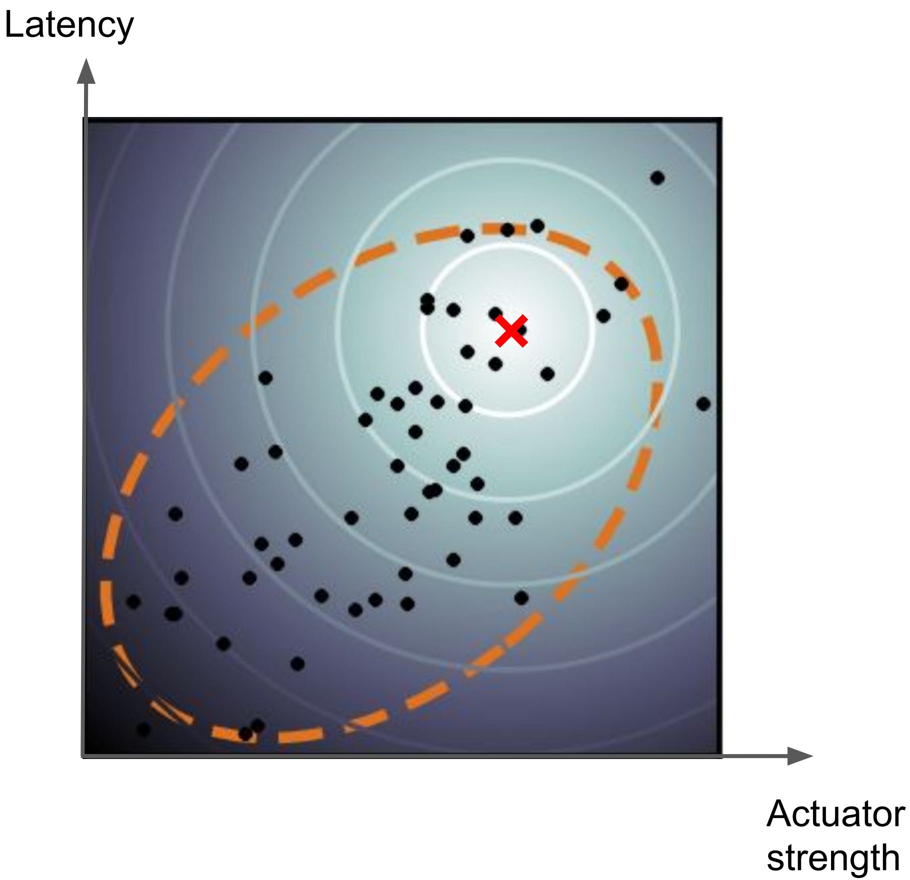
Real trajectory:

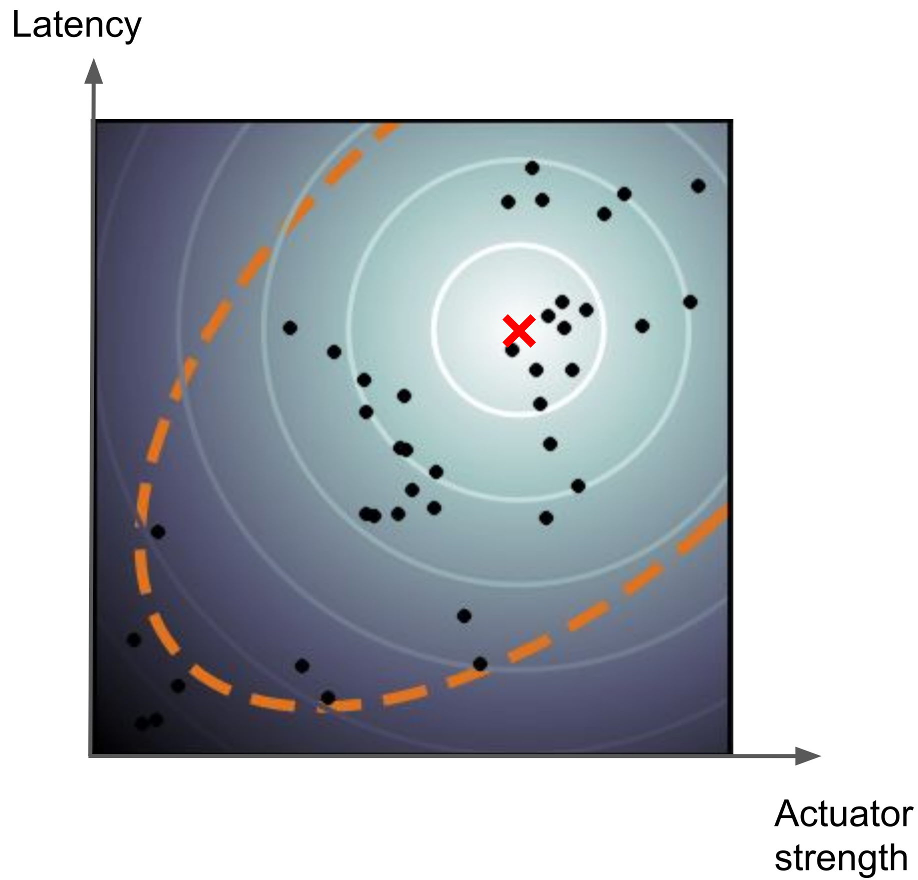


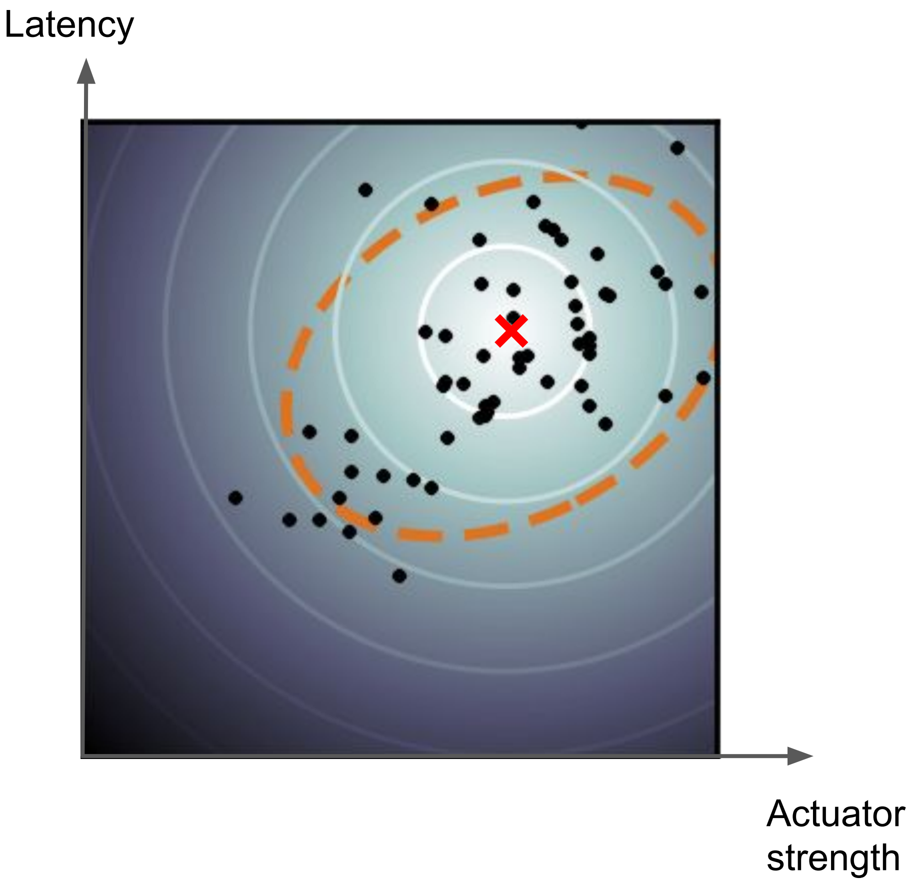


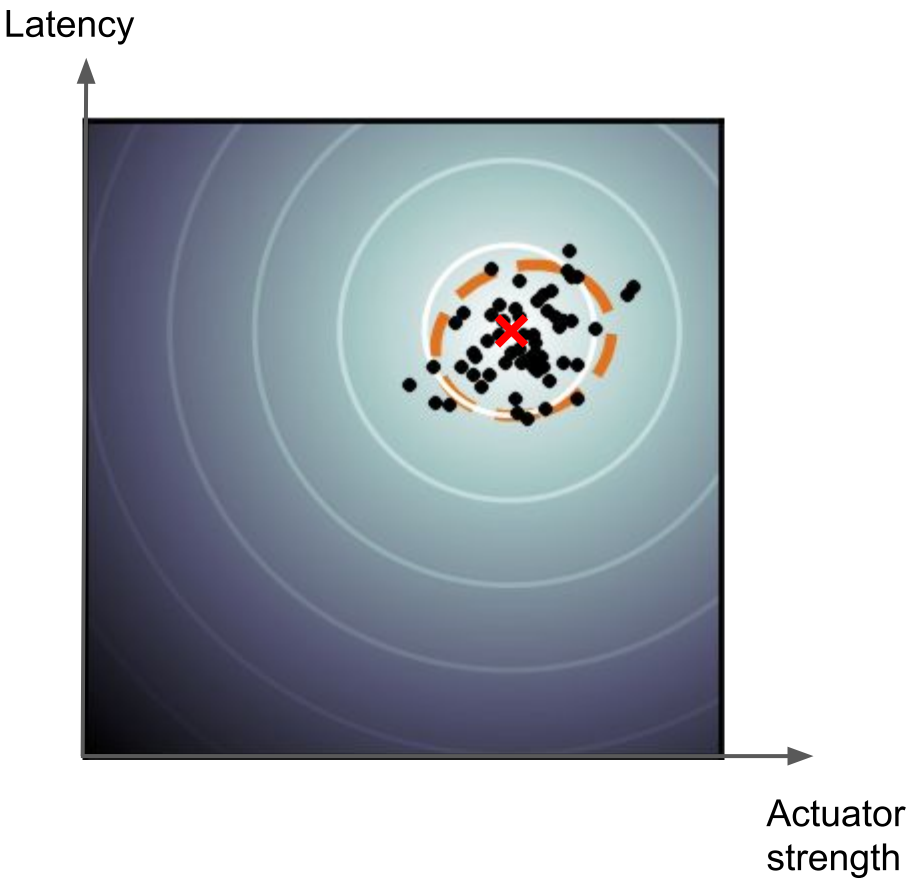
Loss: $\frac{1}{n}\sum_{i=1}^{n}\int_{0}^{T+1} ||\tilde{\mathbf{q}}_i(t) - \mathbf{q}_i(t;\boldsymbol{\theta})||^2_{\mathbf{W}} \mathrm{d}t$

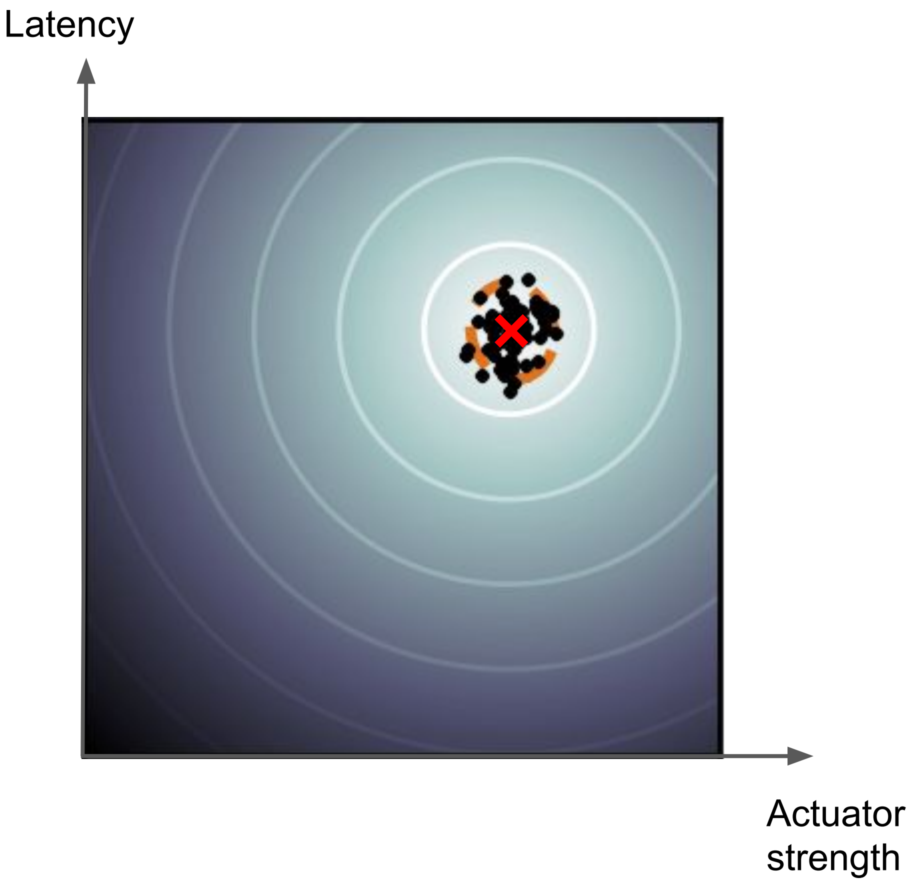

Latency
Actuator strength

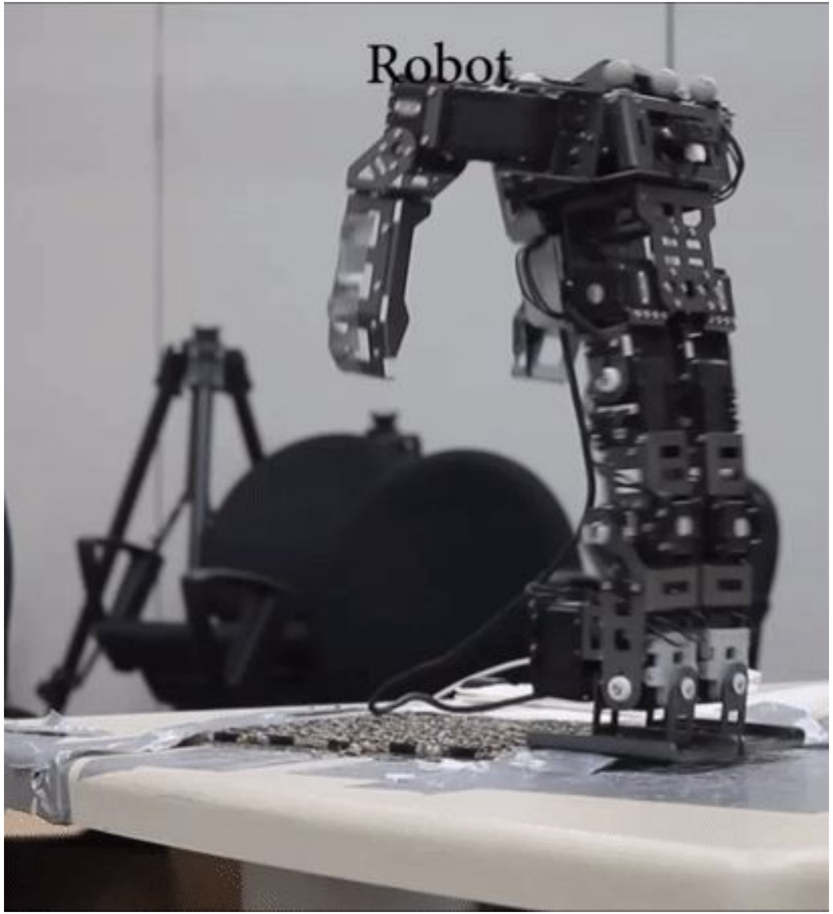

Latency
Actuator strength


Latency
Actuator
strength


Latency
Actuator strength

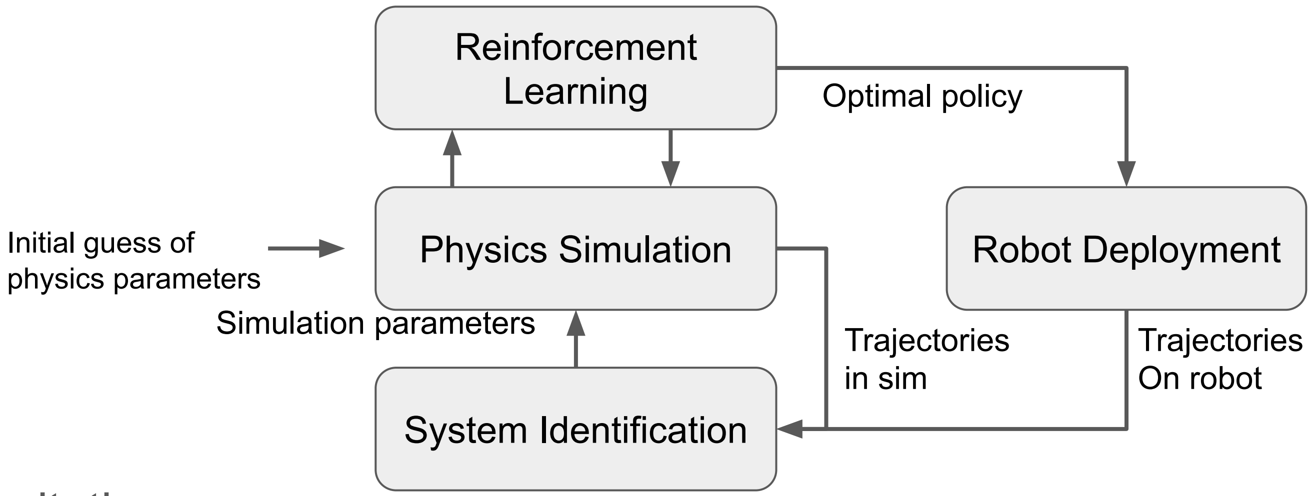

Latency
Actuator
strength


Latency
Actuator strength



[Simulation-based design of dynamic controllers for humanoid balancing, IROS 2016]

Automatic System Identification



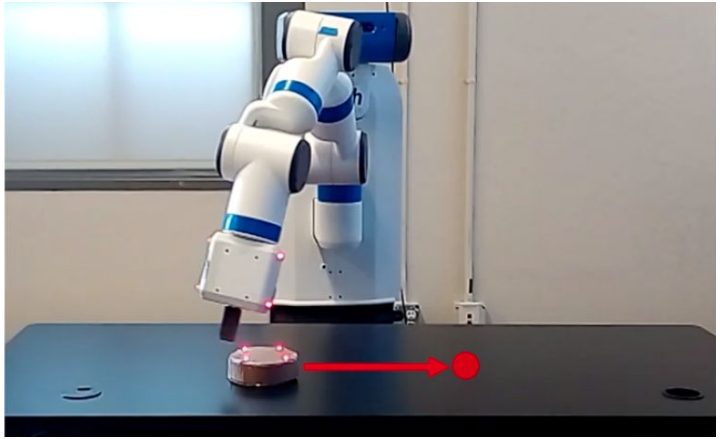
- Limitations
  - Manual selection of physical parameters needed
  - Do not work if sim and real trajectory diverge too quickly
  - Not account for unmodeled dynamics
  - Physical parameters overfit

Domain Randomization

Domain Randomization

- Original objective: reward maximization

$$\mathbb{E}_{\tau \sim p(\tau|\pi)} \left[\sum_{t=0}^{T-1} r(s_t, a_t) \right]$$

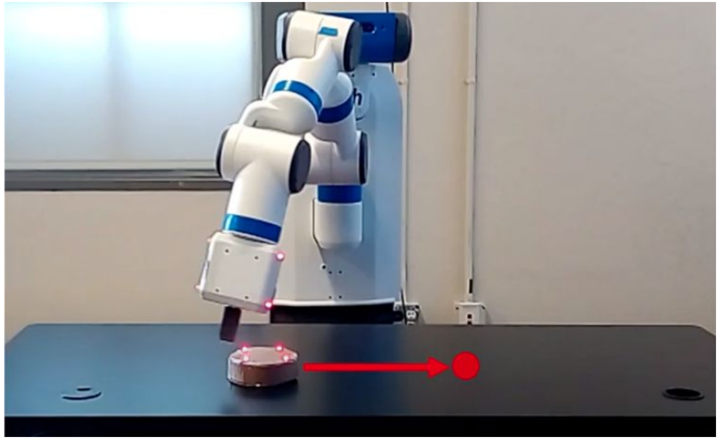


[Sim-to-Real Transfer of Robotic Control with Dynamics Randomization, ICRA 2018]

Domain Randomization

- Original objective: reward maximization

$$\mathbb{E}_{\tau \sim p(\tau|\pi)} \left[\sum_{t=0}^{T-1} r(s_t, a_t) \right]$$



- New objective with domain randomization

$$\boxed{\mathop{\mathbb{E}}_{\mu \sim \rho_\mu}} \left[\mathbb{E}_{\tau \sim p(\tau|\pi \boxed{,\mu})} \left[\sum_{t=0}^{T-1} r(s_t, a_t) \right] \right]$$

Physical parameters

Parameter	Range
Link Mass	$[0.25, 4] \times$ default mass of each link
Joint Damping	$[0.2, 20] \times$ default damping of each joint
Puck Mass	$[0.1, 0.4] kg$
Puck Friction	$[0.1, 5]$
Puck Damping	$[0.01, 0.2] Ns/m$
Table Height	$[0.73, 0.77] m$
Controller Gains	$[0.5, 2] \times$ default gains
Action Timestep λ	$[125, 1000] s^{-1}$

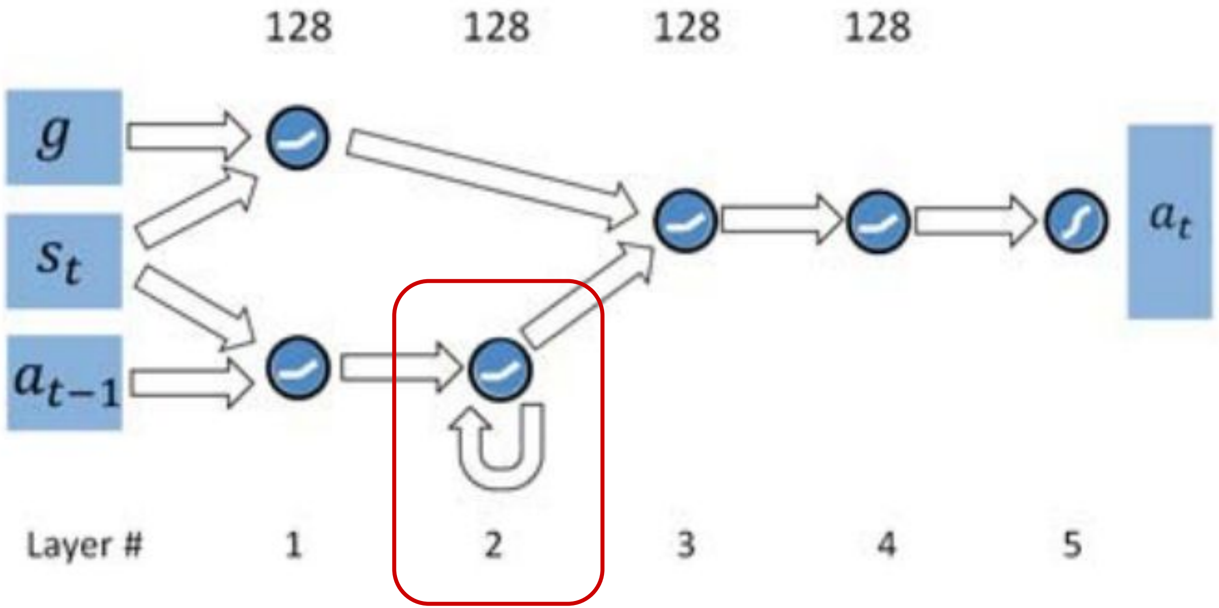
[Sim-to-Real Transfer of Robotic Control with Dynamics Randomization, ICRA 2018]

our method

no randomization during training

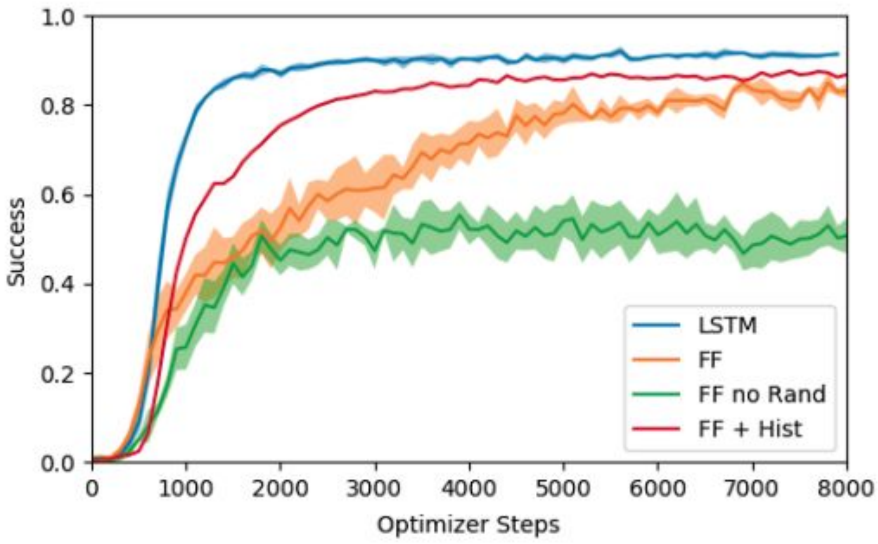
[Sim-to-Real Transfer of Robotic Control with Dynamics Randomization, ICRA 2018]

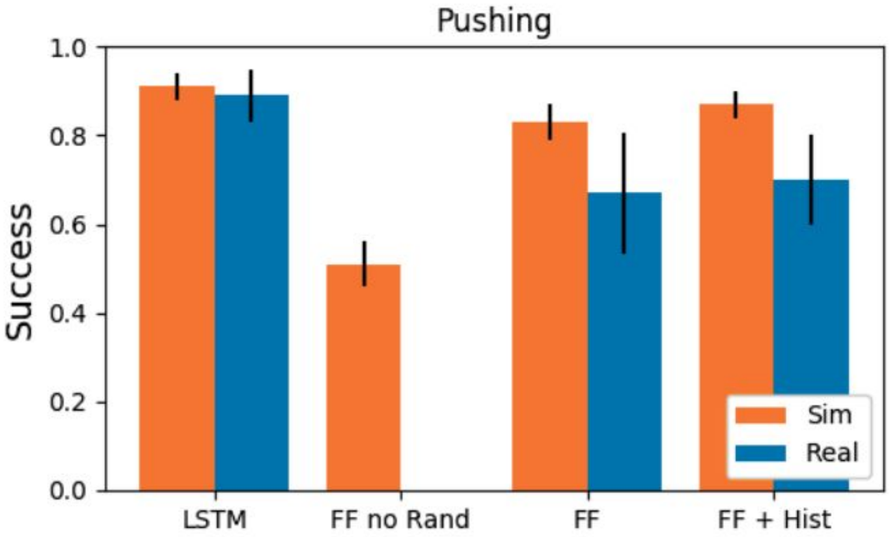
Memory (LSTM) in sim-to-real



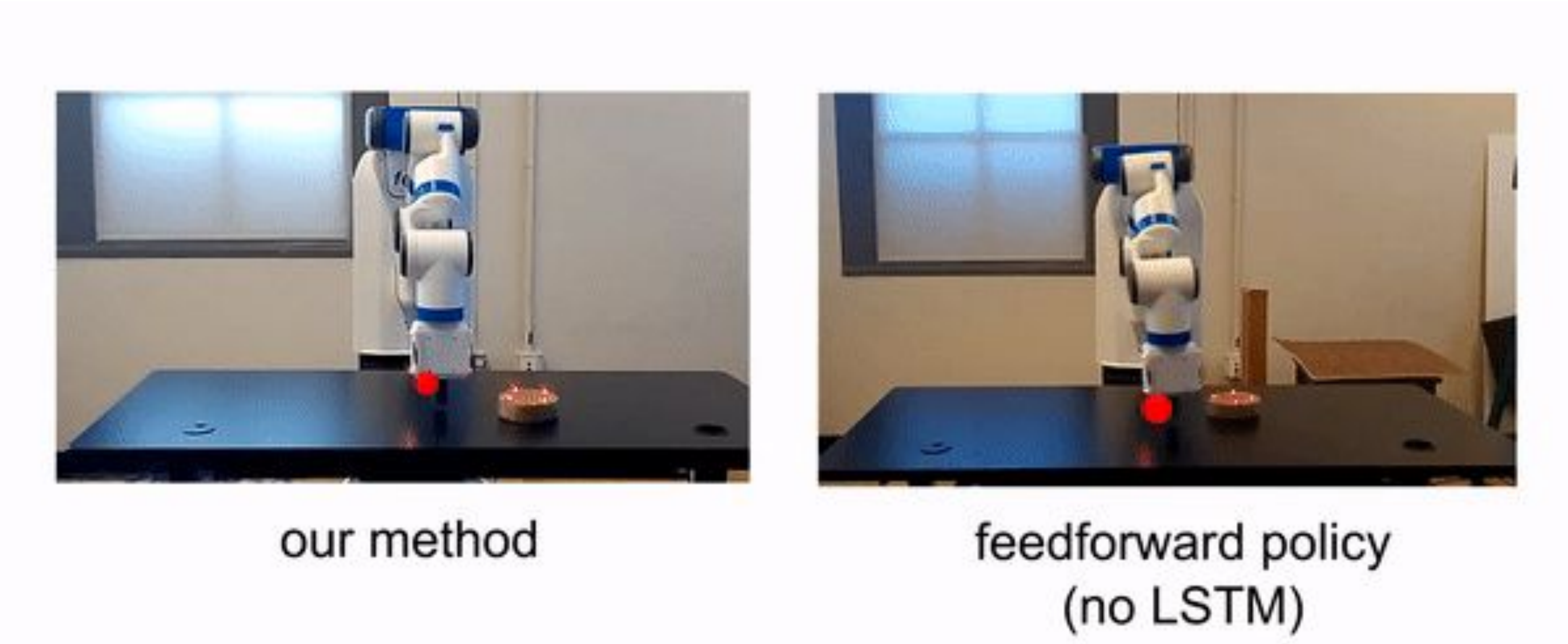
[Sim-to-Real Transfer of Robotic Control with Dynamics Randomization, ICRA 2018]

Memory (LSTM) in sim-to-real





[Sim-to-Real Transfer of Robotic Control with Dynamics Randomization, ICRA 2018]

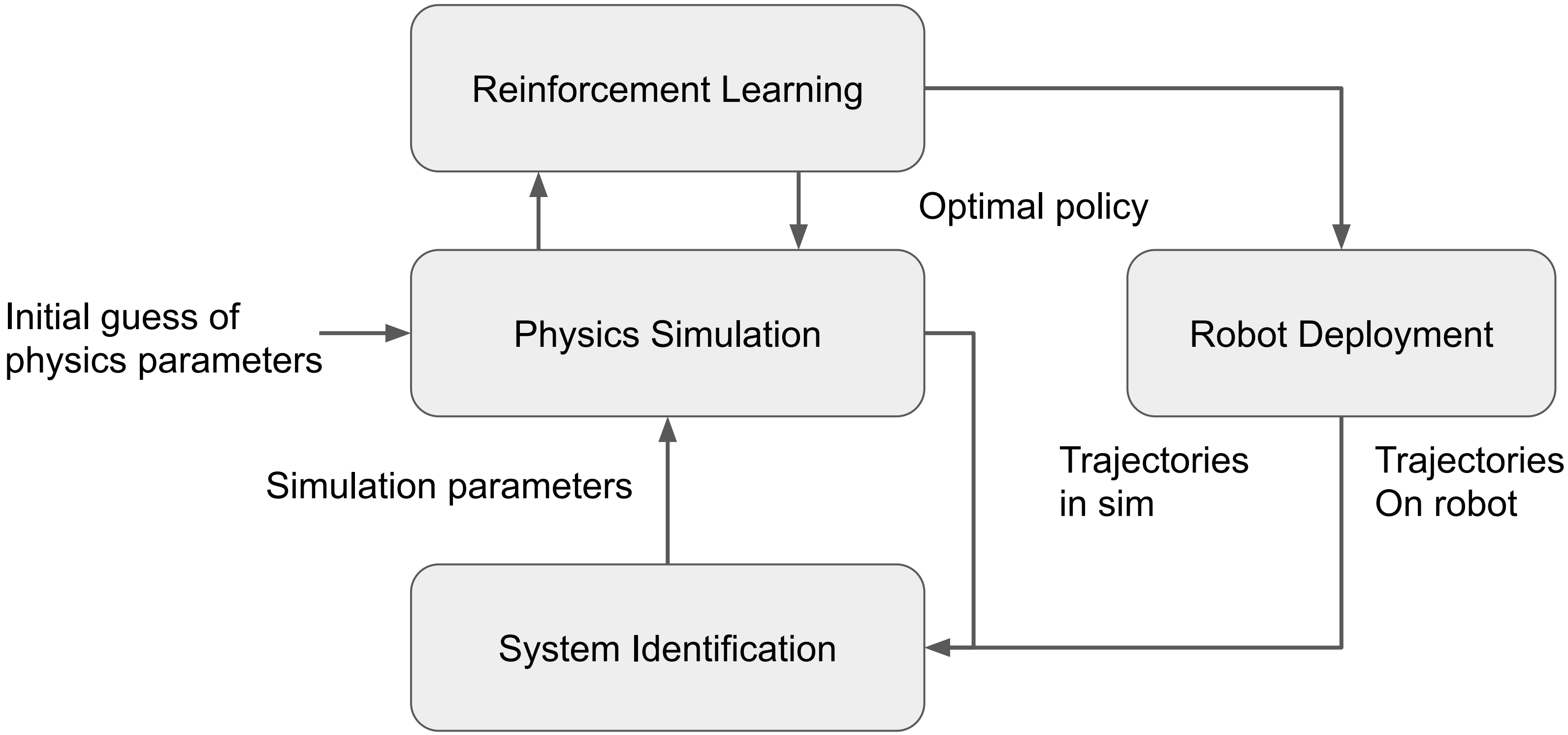


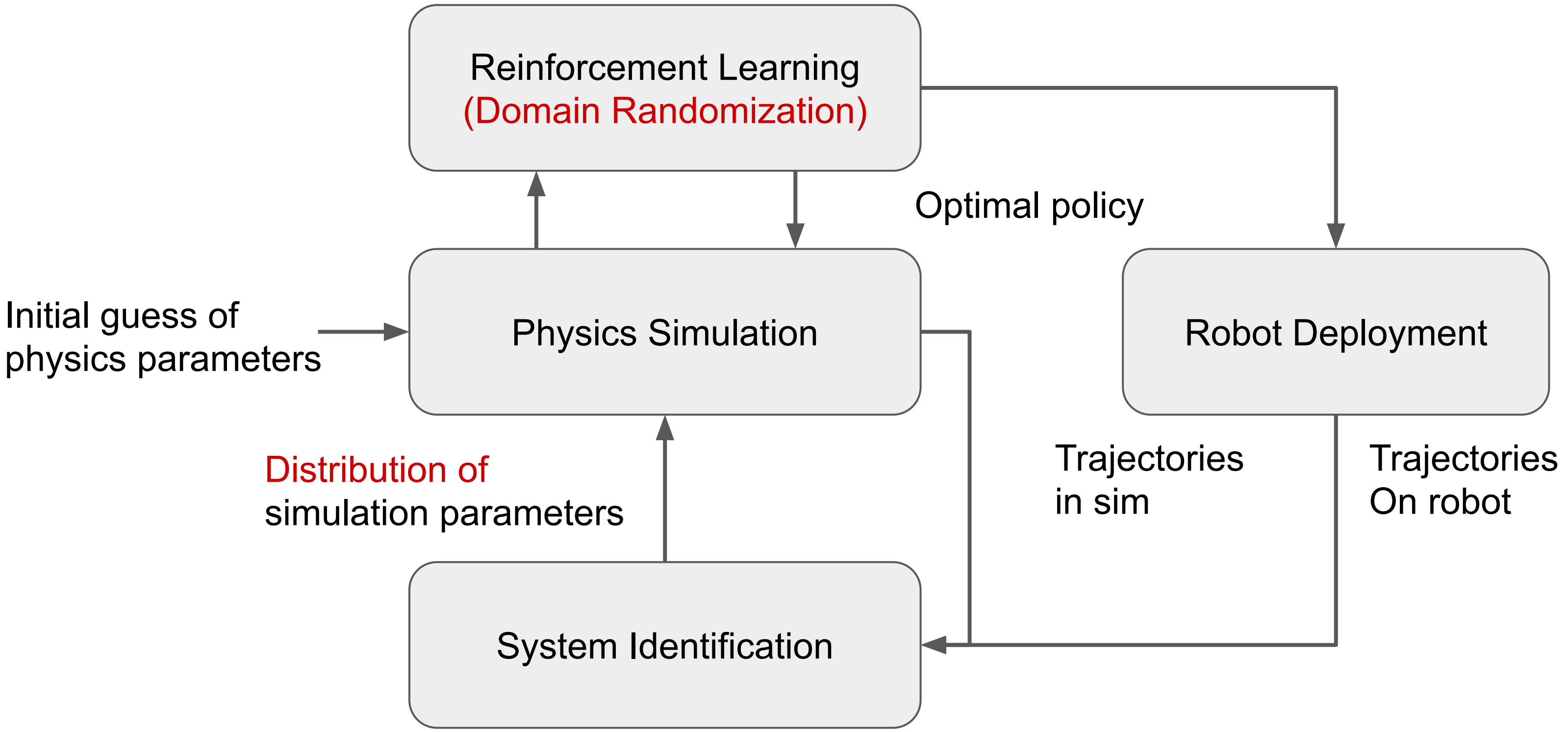
- Limitations
  - Trade optimality for robustness
  - Careful tuning needed for the range of randomization

[Sim-to-Real Transfer of Robotic Control with Dynamics Randomization, ICRA 2018]

Can we use a small amount of real data to mitigate these limitations?

- Limitations
  - Trade optimality for robustness
  - Careful tuning needed for the range of randomization


Reinforcement Learning
Optimal policy
Initial guess of
physics parameters
Physics Simulation
Robot Deployment
Trajectories
in sim
Trajectories
On robot
Simulation parameters
System Identification

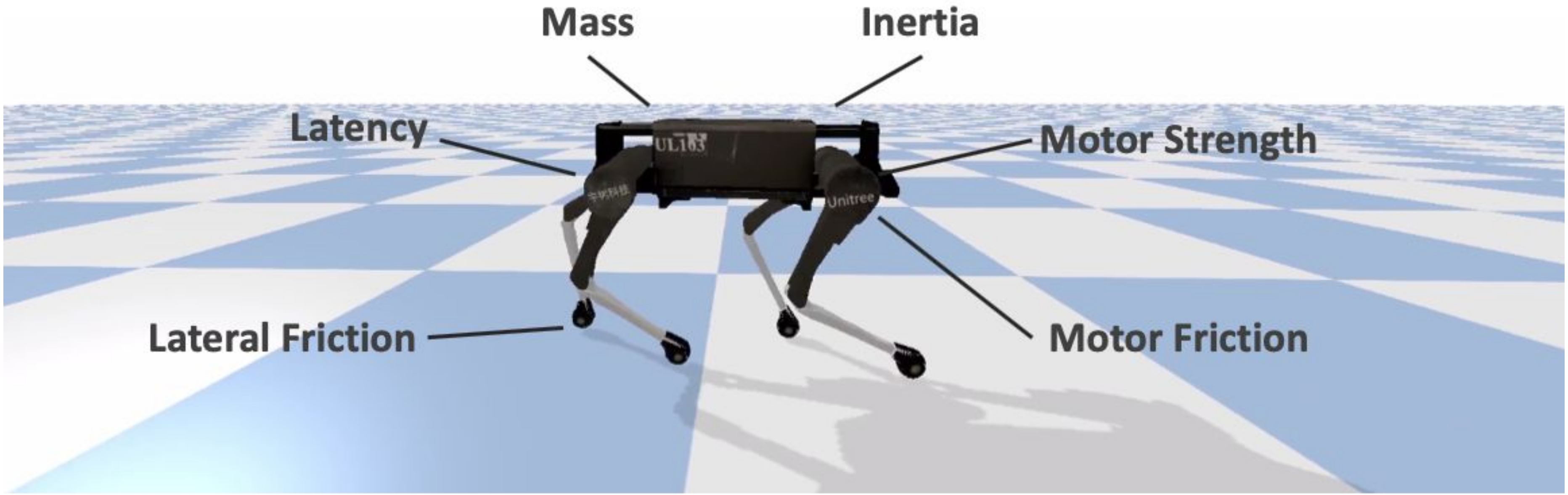


[Closing the Sim-to-Real Loop: Adapting Simulation Randomization with Real World Experience, ICRA 2019]

[Closing the Sim-to-Real Loop: Adapting Simulation Randomization with Real World Experience, ICRA 2019]

Domain Adaptation

Domain Adaptation



[Sim-to-Real Transfer for Biped Locomotion, IROS 2019]

[Learning Agile Robotic Locomotion Skills by Imitating Animals, RSS 2020]

Domain Adaptation

$$\mu = \begin{bmatrix} \text{Mass} \\ \text{Inertia} \\ \text{Motor Strength} \\ \text{Motor Friction} \\ \text{Latency} \\ \text{Lateral Friction} \\ \text{Etc...} \end{bmatrix}$$

[Sim-to-Real Transfer for Biped Locomotion, IROS 2019]

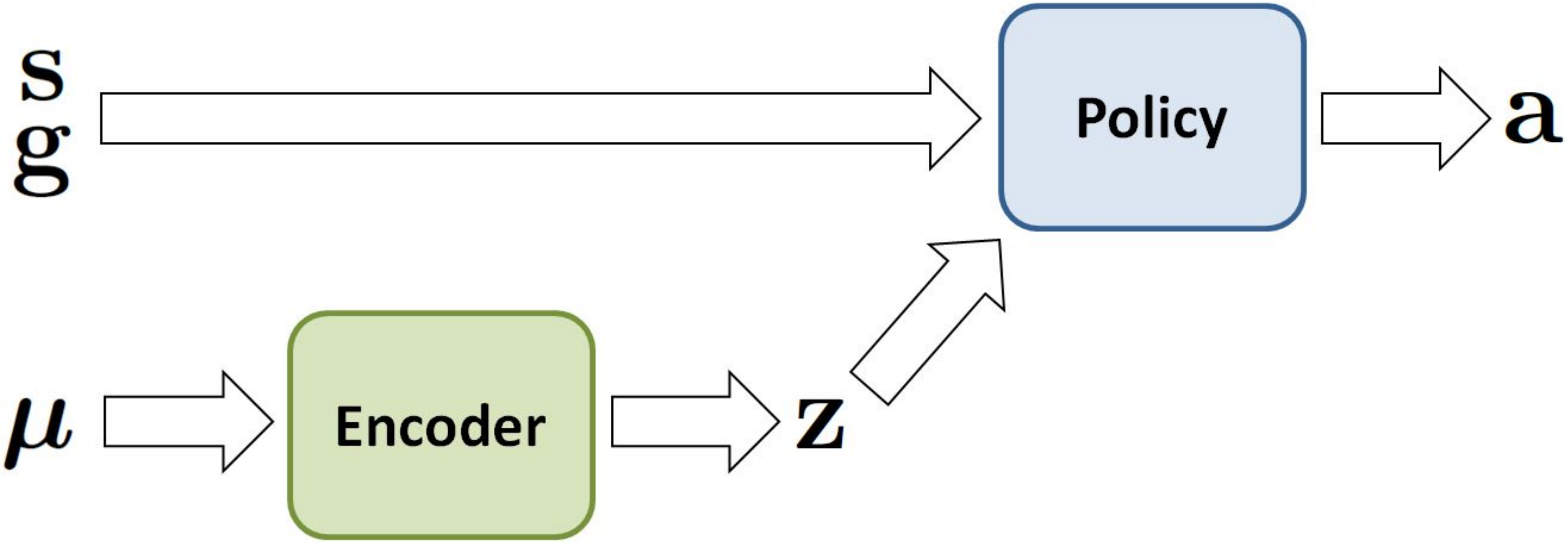
[Learning Agile Robotic Locomotion Skills by Imitating Animals, RSS 2020]

Domain Adaptation

[Sim-to-Real Transfer for Biped Locomotion, IROS 2019]

[Learning Agile Robotic Locomotion Skills by Imitating Animals, RSS 2020]

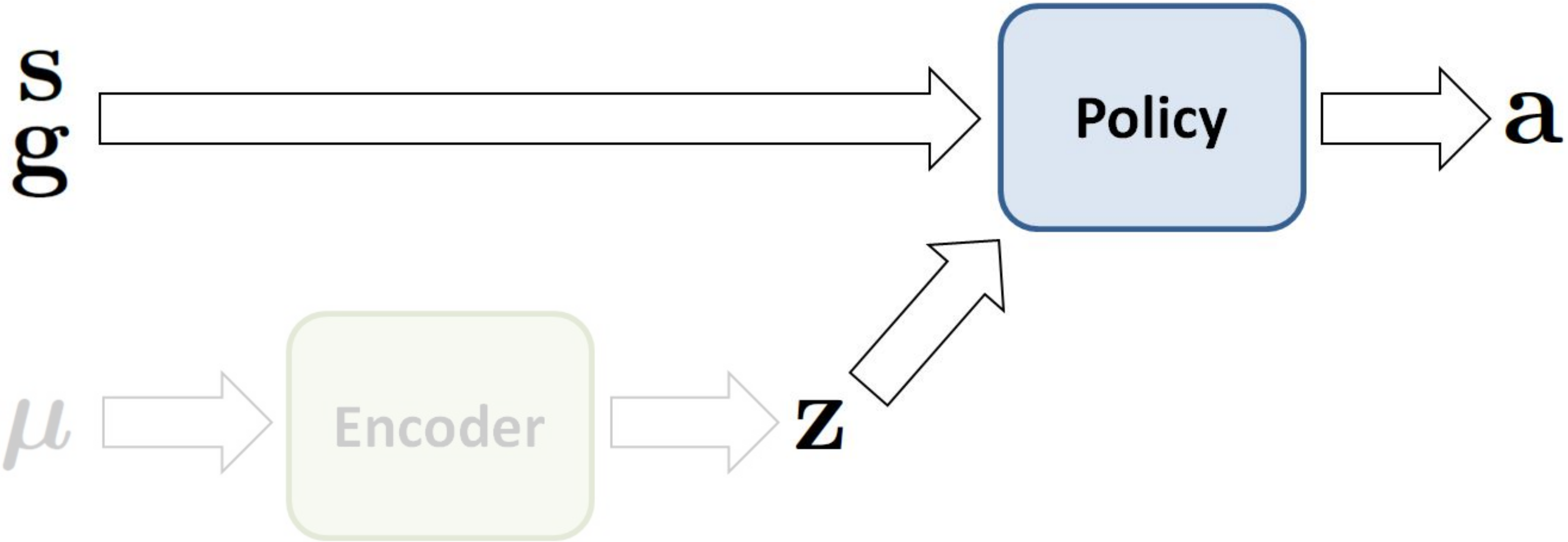
Domain Adaptation



[Sim-to-Real Transfer for Biped Locomotion, IROS 2019]

[Learning Agile Robotic Locomotion Skills by Imitating Animals, RSS 2020]

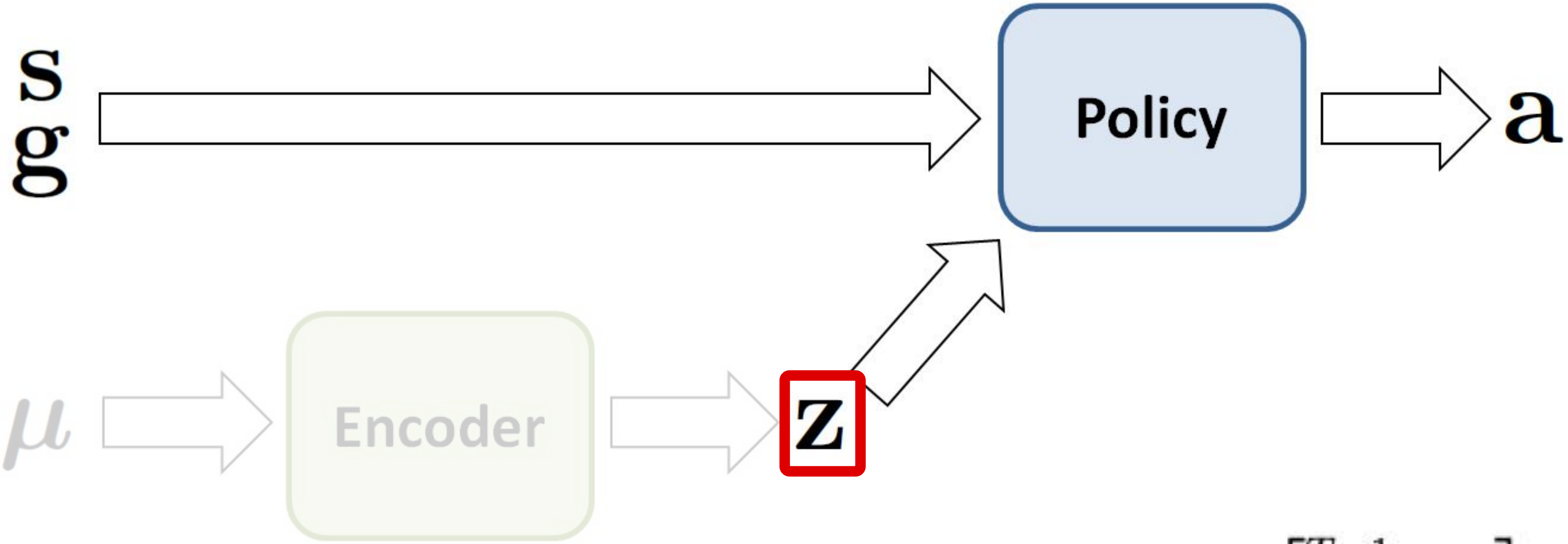
Domain Adaptation



[Sim-to-Real Transfer for Biped Locomotion, IROS 2019]

[Learning Agile Robotic Locomotion Skills by Imitating Animals, RSS 2020]

Domain Adaptation



$$\mathbf{z}^* = \arg\max_{\mathbf{z}} \; \mathbb{E}_{\tau \sim p^*(\tau|\pi,\mathbf{z})} \left[\sum_{t=0}^{T-1} \gamma^t r_t \right]$$

[Sim-to-Real Transfer for Biped Locomotion, IROS 2019]

[Learning Agile Robotic Locomotion Skills by Imitating Animals, RSS 2020]

Domain Adaptation vs. Domain Randomization

Dog Pace

No Randomization

Randomization

Domain Adaptation (Ours)

Domain Adaptation vs. Domain Randomization

Dog Spin

No Randomization | Randomization | Domain Adaptation (Ours)

- Limitations
  - Policy is not updated
  - The latent space may not contain the optimal vector for the real world
  - Performance not necessarily improve with more real data

Meta Learning

MAML for Sim-to-Real Gap

Learn mostly in simulation; quickly adapt to the real world with few real rollouts.

MAML for Sim-to-Real Gap

Learn mostly in simulation; quickly adapt to the real world with few real rollouts.

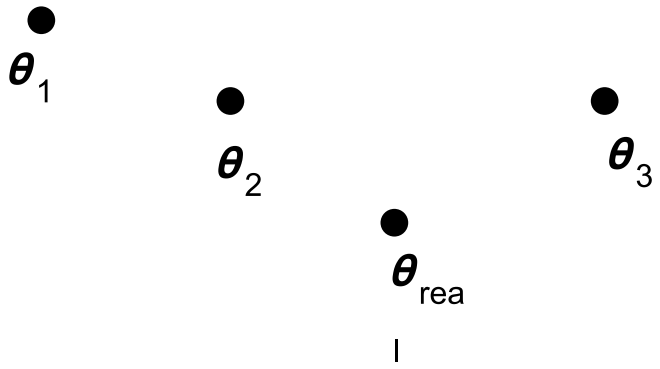
- Real robot: mass = unknown

$\boldsymbol{\theta}_{\mathrm{real}}$

MAML for Sim-to-Real Gap

Learn mostly in simulation; quickly adapt to the real world with few real rollouts.

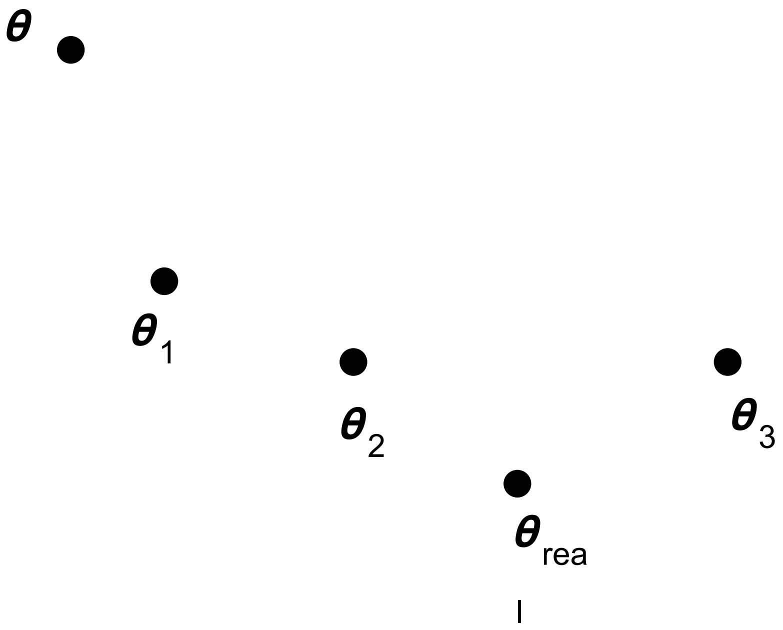
- Real robot: mass = unknown
- Sim task 1: mass = 7.0kg
- Sim task 2: mass = 9.2kg
- Sim task 3: mass = 9.8kg



MAML for Sim-to-Real Gap

Learn mostly in simulation; quickly adapt to the real world with few real rollouts.

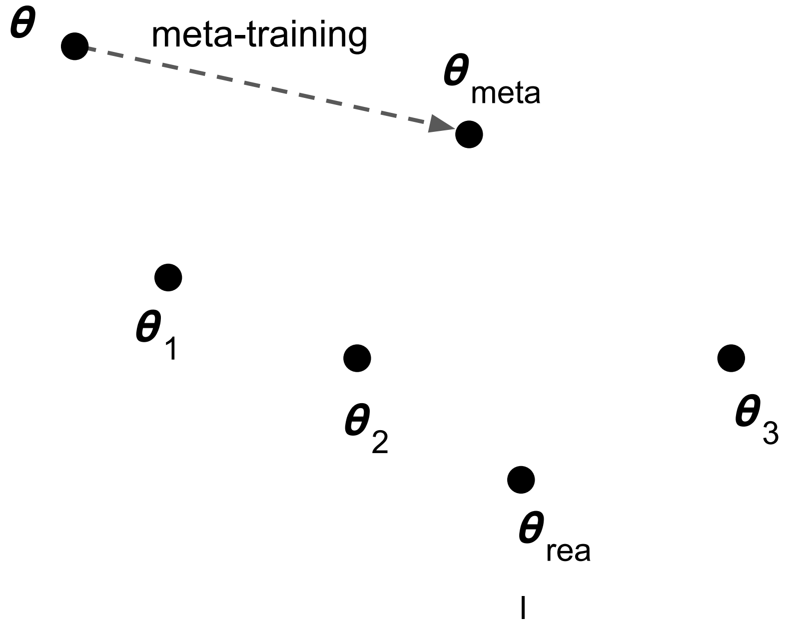
- Real robot: mass = unknown
- Sim task 1: mass = 7.0kg
- Sim task 2: mass = 9.2kg
- Sim task 3: mass = 9.8kg



MAML for Sim-to-Real Gap

Learn mostly in simulation; quickly adapt to the real world with few real rollouts.

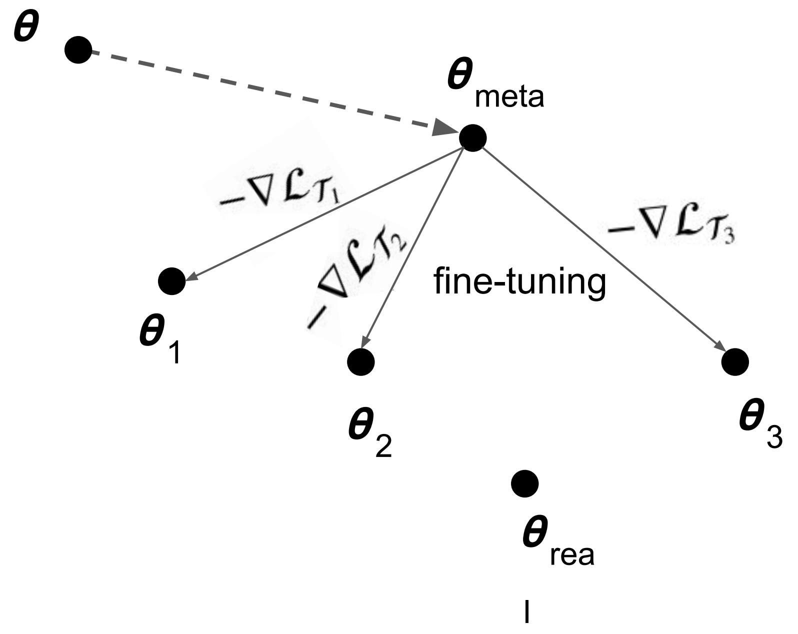
- Real robot: mass = unknown
- Sim task 1: mass = 7.0kg
- Sim task 2: mass = 9.2kg
- Sim task 3: mass = 9.8kg



MAML for Sim-to-Real Gap

Learn mostly in simulation; quickly adapt to the real world with few real rollouts.

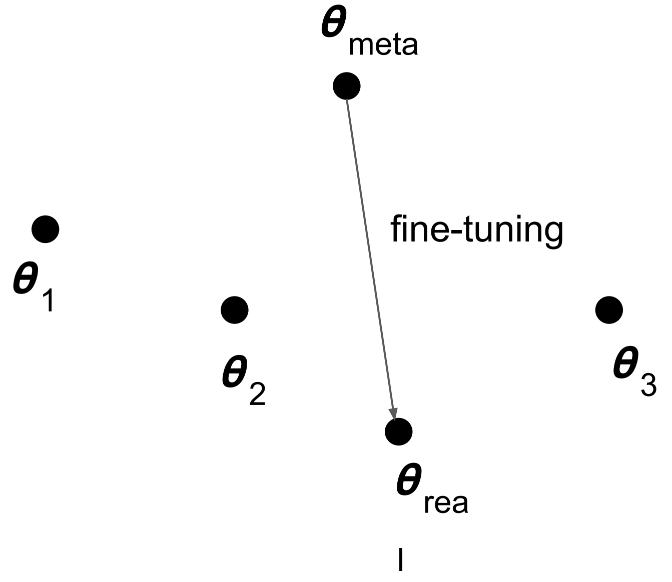
- Real robot: mass = unknown
- Sim task 1: mass = 7.0kg
- Sim task 2: mass = 9.2kg
- Sim task 3: mass = 9.8kg



MAML for Sim-to-Real Gap

Learn mostly in simulation; quickly adapt to the real world with few real rollouts.

- Real robot: mass = unknown
- Sim task 1: mass = 7.0kg
- Sim task 2: mass = 9.2kg
- Sim task 3: mass = 9.8kg



MAML for Sim-to-Real Gap

- Fine-tuning

$$\theta_i' = \theta - \alpha \nabla_\theta \mathcal{L}_{\mathcal{T}_i}(f_\theta)$$

- Meta-training

$$\min_\theta \sum_{\mathcal{T}_i \sim p(\mathcal{T})} \mathcal{L}_{\mathcal{T}_i}(f_{\theta_i'})$$

Algorithm 1 Model-Agnostic Meta-Learning

Require: $p(\mathcal{T})$: distribution over tasks
Require: α, β: step size hyperparameters

```
randomly initialize θ
while not done do
    Sample batch of tasks T_i ~ p(T)
    for all T_i do
        Evaluate ∇_θ L_{T_i}(f_θ) with respect to K examples
        Compute adapted parameters with gradient descent: θ'_i = θ − α∇_θ L_{T_i}(f_θ)
    end for
    Update θ ← θ − β∇_θ Σ_{T_i~p(T)} L_{T_i}(f_{θ'_i})
end while
```

MAML for Sim-to-Real Gap

- Fine-tuning

$$\theta_i' = \theta - \alpha \nabla_\theta \mathcal{L}_{\mathcal{T}_i}(f_\theta)$$

- Meta-training

$$\min_\theta \sum_{\mathcal{T}_i \sim p(\mathcal{T})} \mathcal{L}_{\mathcal{T}_i}(f_{\theta_i'})$$

Algorithm 1 Model-Agnostic Meta-Learning

Require: $p(\mathcal{T})$: distribution over tasks
Require: α, β: step size hyperparameters

randomly initialize θ
while not done **do**
  Sample batch of tasks $\mathcal{T}_i \sim p(\mathcal{T})$
  for all $\mathcal{T}_i$ **do**
    Evaluate $\nabla_\theta \mathcal{L}_{\mathcal{T}_i}(f_\theta)$ with respect to K examples
    Compute adapted parameters with gradient descent: $\theta_i' = \theta - \alpha \nabla_\theta \mathcal{L}_{\mathcal{T}_i}(f_\theta)$
  end for
  Update $\theta \leftarrow \theta - \beta \nabla_\theta \sum_{\mathcal{T}_i \sim p(\mathcal{T})} \mathcal{L}_{\mathcal{T}_i}(f_{\theta_i'})$
end while

MAML for Sim-to-Real Gap

- Fine-tuning

$$\theta_i' = \theta - \alpha \nabla_\theta \mathcal{L}_{\mathcal{T}_i}(f_\theta)$$

- Meta-training

$$\min_\theta \sum_{\mathcal{T}_i \sim p(\mathcal{T})} \mathcal{L}_{\mathcal{T}_i}(f_{\theta_i'})$$

Algorithm 1 Model-Agnostic Meta-Learning

Require: $p(\mathcal{T})$: distribution over tasks
Require: α, β: step size hyperparameters
1: randomly initialize θ
2: **while** not done **do**
3: Sample batch of tasks $\mathcal{T}_i \sim p(\mathcal{T})$
4: **for all** $\mathcal{T}_i$ **do**
5: Evaluate $\nabla_\theta \mathcal{L}_{\mathcal{T}_i}(f_\theta)$ with respect to K examples
6: Compute adapted parameters with gradient descent: $\theta_i' = \theta - \alpha \nabla_\theta \mathcal{L}_{\mathcal{T}_i}(f_\theta)$
7: **end for**
8: Update $\theta \leftarrow \theta - \beta \nabla_\theta \sum_{\mathcal{T}_i \sim p(\mathcal{T})} \mathcal{L}_{\mathcal{T}_i}(f_{\theta_i'})$
9: **end while**

Gradient Estimation using Evolution Strategy (ES)

$$\nabla_\theta \tilde{f}_\sigma(\theta) = \frac{1}{\sigma}\mathbb{E}_{\underbrace{\mathbf{g}\sim\mathcal{N}(0,\mathbf{I}_d)}_{\text{Perturbation direction}}}[\underbrace{f(\theta+\sigma\mathbf{g})}_{\text{Return of perturbed policy}}\mathbf{g}]$$

Gradient Estimation using Evolution Strategy (ES)

$$\nabla_\theta \tilde{f}_\sigma(\theta) = \underbrace{\frac{1}{\sigma}\mathbb{E}_{\mathbf{g}\sim\mathcal{N}(0,\mathbf{I}_d)}[f(\theta+\sigma\mathbf{g})\mathbf{g}]}_{\text{Estimated policy gradient}}$$

ES-MAML for Reality Gap

Algorithm 1 Model-Agnostic Meta-Learning

Require: $p(\mathcal{T})$: distribution over tasks

Require: α, β: step size hyperparameters

```
randomly initialize θ
while not done do
    Sample batch of tasks T_i ~ p(T)
    for all T_i do
        Evaluate ∇_θ L_{T_i}(f_θ) with respect to K examples
        Compute adapted parameters with gradient descent: θ'_i = θ − α∇_θ L_{T_i}(f_θ)
    end for
    Update θ ← θ − β∇_θ Σ_{T_i~p(T)} L_{T_i}(f_{θ'_i})
end while
```

Data: initial policy θ_0, adaptation step size α, meta step size β, number of queries K

```
for t = 0, 1, ... do
    Sample n tasks T_1, ..., T_n and iid vectors g_1, ..., g_n ~ N(0, I);
    foreach (T_i, g_i) do
        d^(i) ← ESGRAD(f^{T_i}, θ_t + σg_i, K, σ);
        θ_t^(i) ← θ_t + σg_i + αd^(i);
        v_i ← f^{T_i}(θ_t^(i));
    end
    θ_{t+1} ← θ_t + (β/(σn)) Σ_{i=1}^{n} v_i g_i;
end
```

ES-MAML for Reality Gap

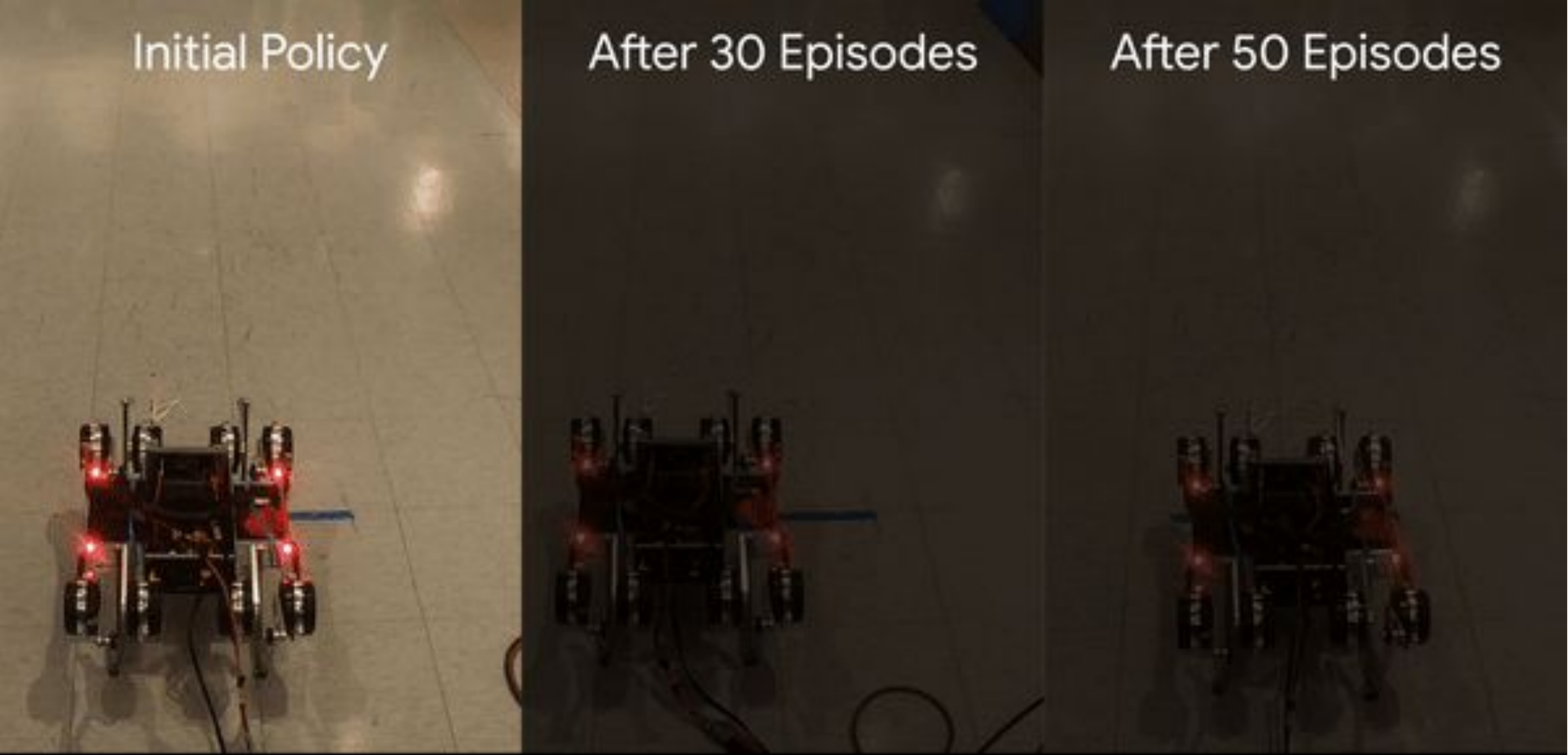
Algorithm 1 Model-Agnostic Meta-Learning

Require: $p(\mathcal{T})$: distribution over tasks
Require: α, β: step size hyperparameters

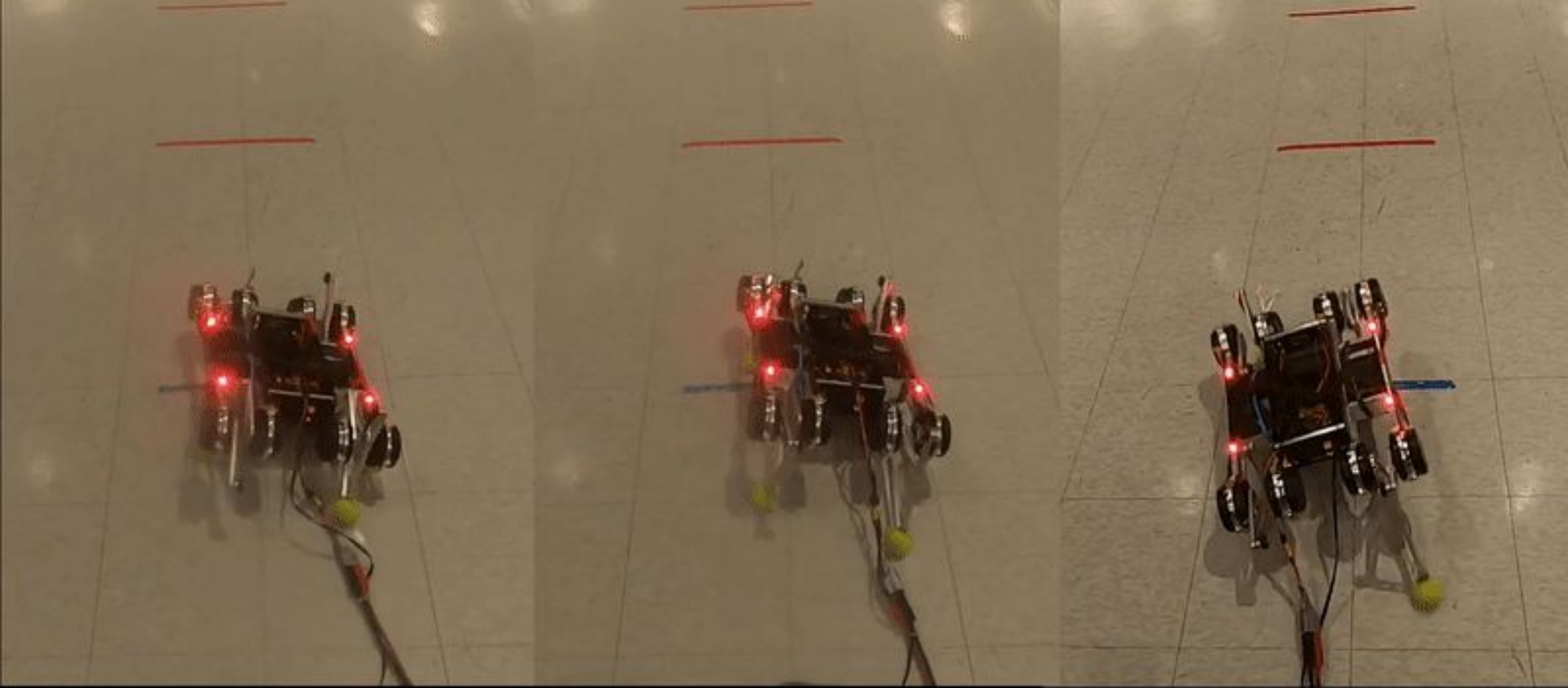
```
randomly initialize θ
while not done do
    Sample batch of tasks T_i ~ p(T)
    for all T_i do
        Evaluate ∇_θ L_{T_i}(f_θ) with respect to K examples
        Compute adapted parameters with gradient descent: θ'_i = θ − α∇_θ L_{T_i}(f_θ)
    end for
    Update θ ← θ − β∇_θ Σ_{T_i~p(T)} L_{T_i}(f_{θ'_i})
end while
```

Data: initial policy θ_0, adaptation step size α, meta step size β, number of queries K

```
for t = 0, 1, ... do
    Sample n tasks T_1, ..., T_n and iid vectors g_1, ..., g_n ~ N(0, I);
    foreach (T_i, g_i) do
        d^(i) ← ESGRAD(f^{T_i}, θ_t + σg_i, K, σ);
        θ_t^(i) ← θ_t + σg_i + αd^(i);
        v_i ← f^{T_i}(θ_t^(i));
    end
    θ_{t+1} ← θ_t + β/(σn) Σ_{i=1}^n v_i g_i;
end
```



The initial policy shifts to the right.



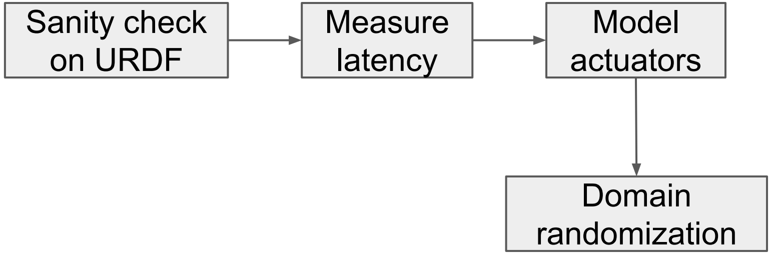
Domain Randomization | PG-MAML | Our Method

Summary

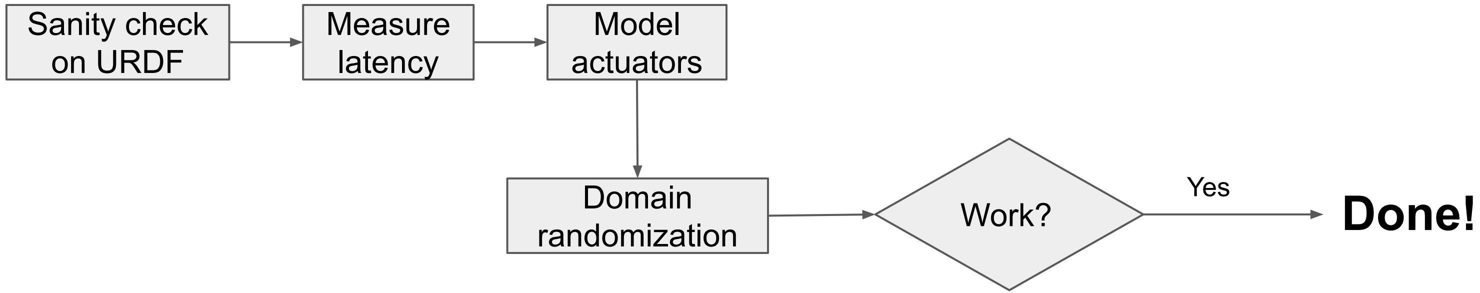
- Improve simulation
  - System identification
    - Always should be the first step for sim-to-real
    - Identified parameters can be reused
    - Require diverse data and careful experiment design
    - Can be tedious
- Improve policy
  - Domain randomization
    - Simple
    - Zero-shot transfer is often possible
    - Trade-off between robustness and optimality
  - Domain adaptation
    - Small amount of real-world data needed
    - Good experience so far in multiple locomotion projects
  - Meta learning
    - Novel and popular research direction

Pipeline for Sim-to-Real

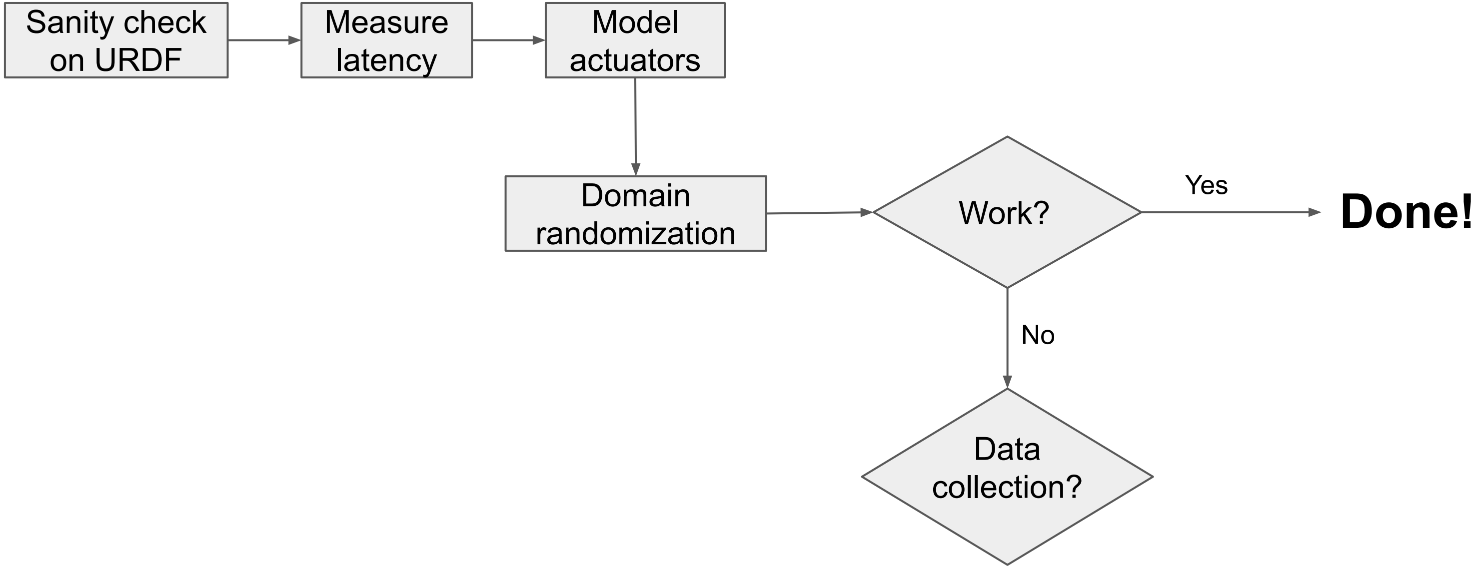
Pipeline for Sim-to-Real



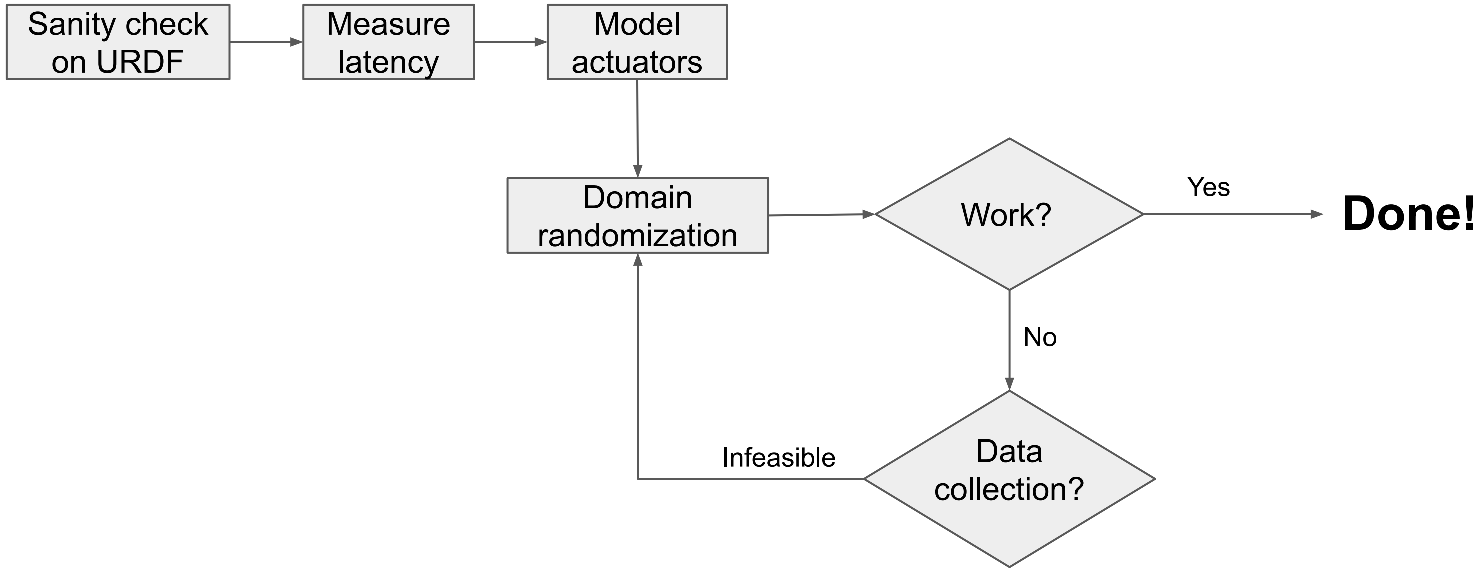
Pipeline for Sim-to-Real



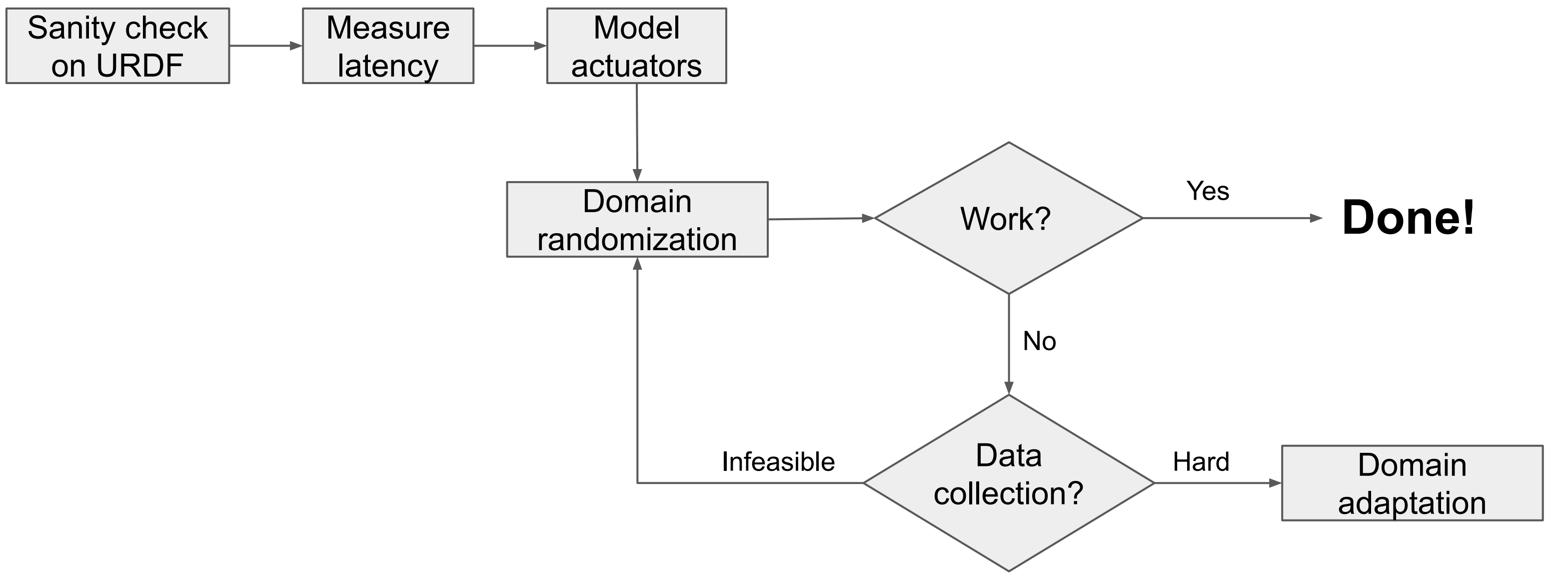
Pipeline for Sim-to-Real



Pipeline for Sim-to-Real



Pipeline for Sim-to-Real



Pipeline for Sim-to-Real