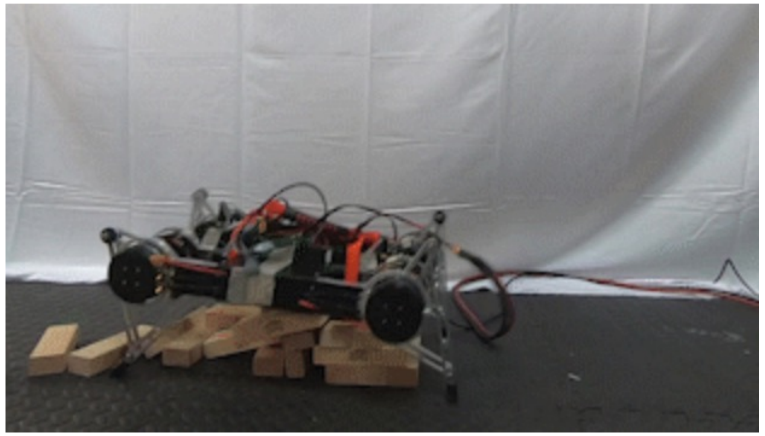
Thesis Proposal: Physics-Aware Robot Learning

Meng Song

6-9-2021

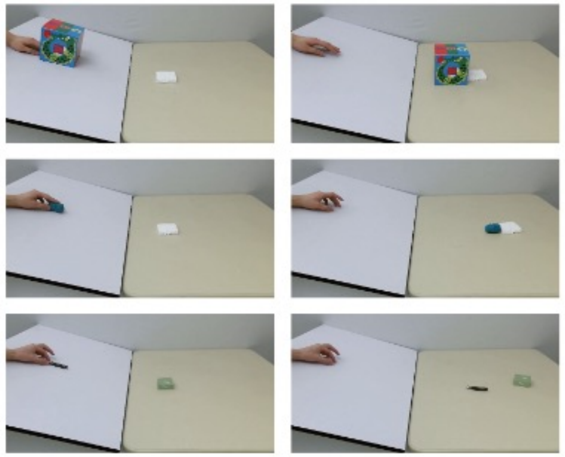
Physical understanding is key ingredient of intelligence

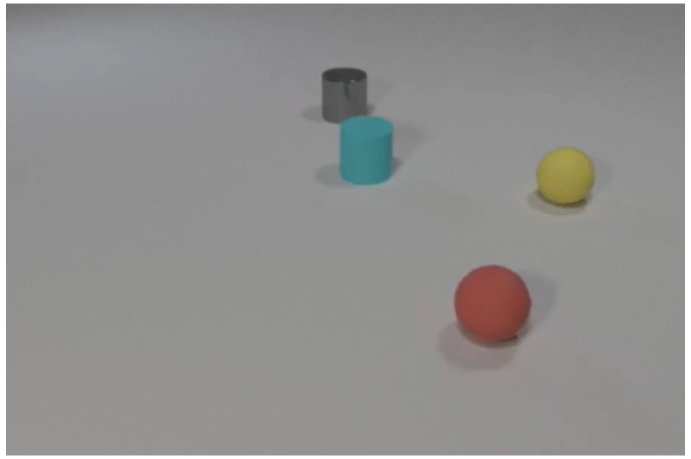


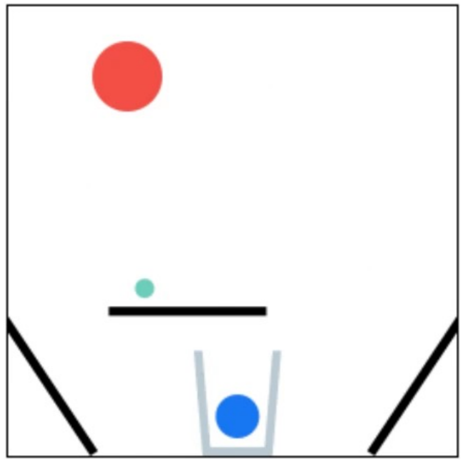
Haarnoja, Ha, Zhou, et al., Learning to Walk via Deep Reinforcement Learning, 2018

Understanding Physics by Observing

- Learn by observing







Wu, Yildirim, Lim, et al., Galileo: Perceiving physical object properties by integrating a physics engine with deep learning, 2015
Bakhtin, Maaten, Johnson, et al., Phyre: A new benchmark for physical reasoning, 2019
Yi, Gan, Li, et al., Clevrer: Collision events for video representation and reasoning, 2020

Understanding Physics by Performing Tasks

- Learn by doing

Understanding Physics by Performing Tasks

- Learn physics by a continuous decision making process driven by a task
  - Interact, get feedback, improve the knowledge, and make decision accordingly.

PC Game Ballance

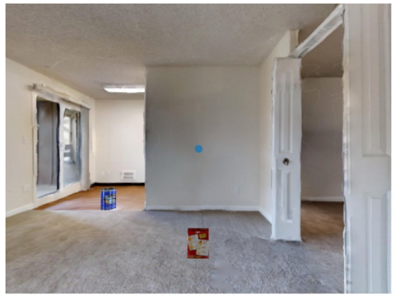
Current robotics simulation environments

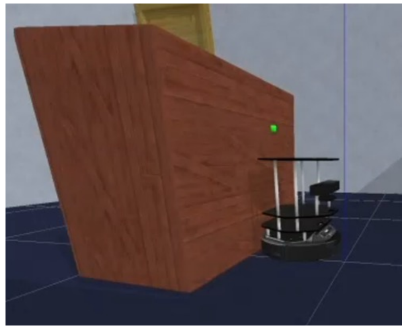
- Navigation
  - Others: Aim to tackle the long-horizon navigation challenges
    - Large-scale indoor scene across multiple rooms
    - Building the environment map
    - Hierarchical planning
    - Discretize the state space into a grid world
    - Lack of physics-driven movements
  - Ours
    - Fine-grained mechanical interactions with joint-space control
    - Restrict tasks to **a single room** scene to avoid hard exploration

B. Shen, F. Xia, C. Li, et al., iGibson: A simulation environment for interactive tasks in large realistic scenes, 2020

Current robotics simulation environments

- Manipulation
  - Others
    - Table-top manipulation
      - Requires gripper and arm
    - Magic pointer
      - Grab and release object in the pixel space
      - Not physics-driven
  - Ours
    - Pushing while moving
      - Unify navigation and object interaction skills
      - Avoid the need for dexterous manipulation

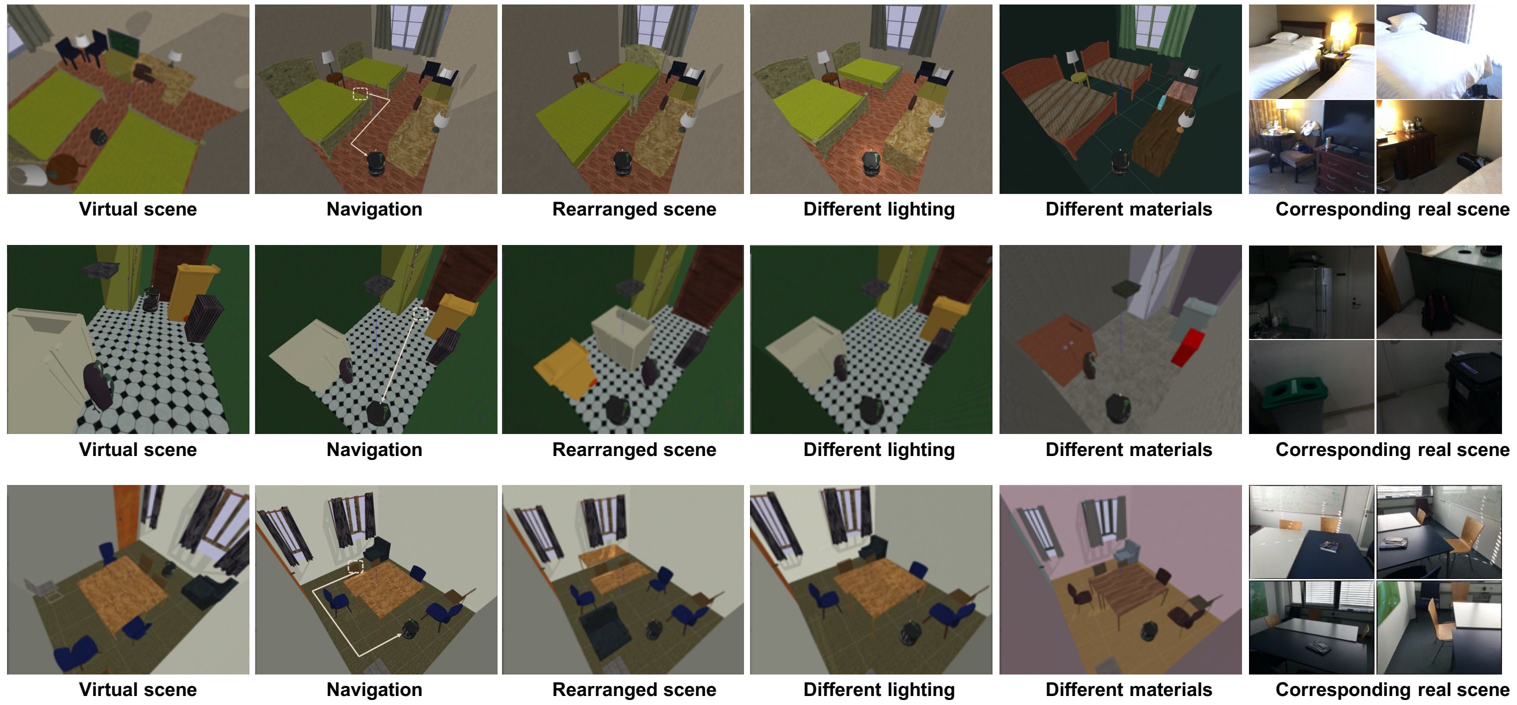




Savva, Kadian, Maksymets, et al., Habitat: A Platform for Embodied AI Research, 2019
Singh, Yu, Yang, et al., COG: Connecting New Skills to Past Experience with Offline Reinforcement Learning, 2020
Chebotar, Hausman, Lu, et al., Actionable Models: Unsupervised Offline Reinforcement Learning of Robotic Skills, 2021

Openrooms

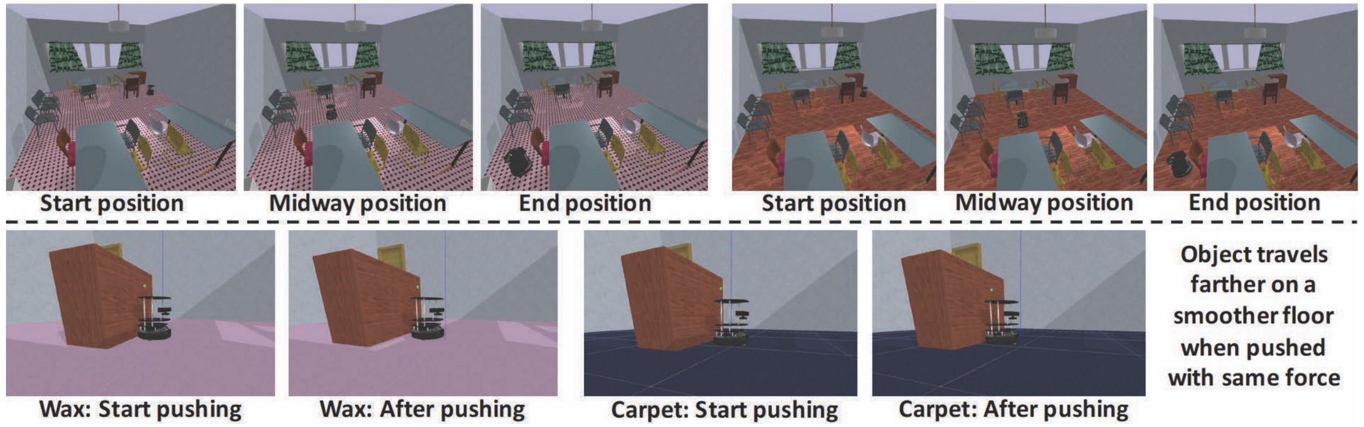
- Unique ground-truth of properties such as materials, friction coefficients, masses, and correspondence to real scenes.



Zhengqin Li, Ting-Wei Yu, Shen Sang, Sarah Wang, **Meng Song**, et al.,
OpenRooms: An End-to-End Open Framework for Photorealistic Indoor Scene Datasets in Conference on Computer Vision and Pattern Recognition (CVPR) [Oral], 2021

Openrooms

- Integrating with physics engine Bullet, enables robot learning in highly physics realistic environments
  - Navigation based object searching
  - Room rearrangement

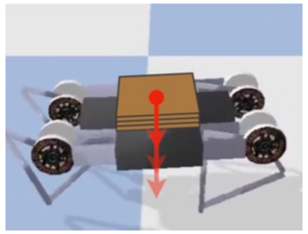


Zhengqin Li, Ting-Wei Yu, Shen Sang, Sarah Wang, **Meng Song**, et al.,
OpenRooms: An End-to-End Open Framework for Photorealistic Indoor Scene Datasets in Conference on Computer Vision and Pattern Recognition (CVPR) [Oral], 2021

Openrooms

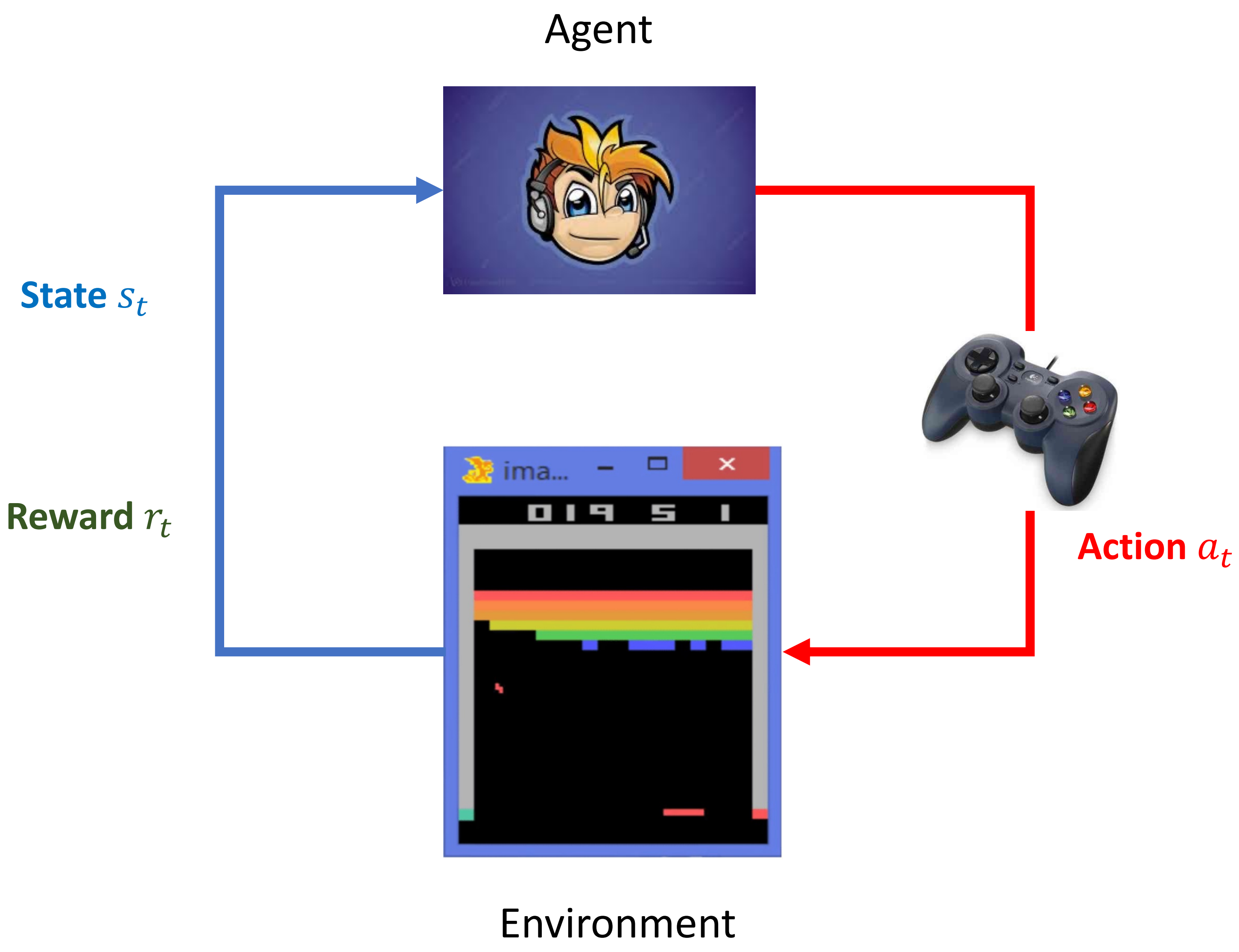
- Transfer learning
  - Transfer skills to similar tasks in environments with different masses, frictions, lightings
  - Learn and adapt in a non-stationary environment with gradually changing dynamics in the real-world setting
    - Different terrains

    - Variable payloads



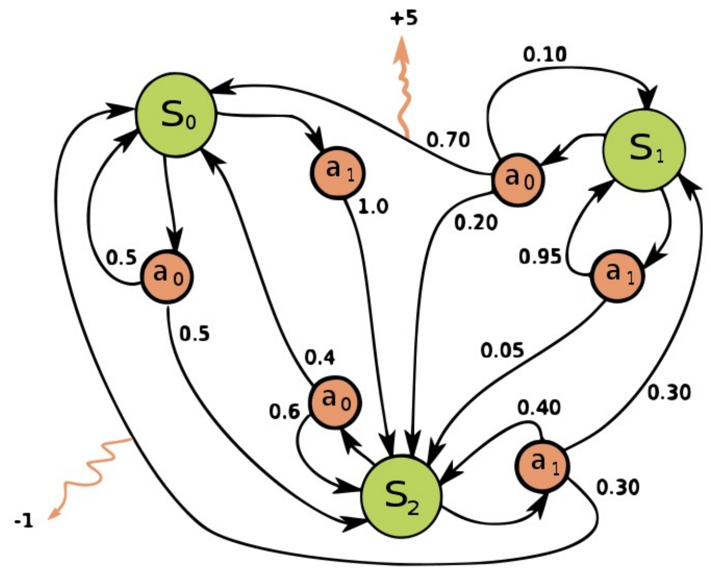
Shah, Eysenbach, Kahn, et al., ViNG: Learning Open-World Navigation with Visual Goals, 2020
Xie, Harrison, Finn, Deep Reinforcement Learning amidst Lifelong Non-Stationarity, 2020

Sequential Decision Making Process



- At each step *t* the **agent**:
  - Takes an action a_t
  - Receives a state s_t
  - Receives a scalar reward r_t
- At each step *t* the **environment**:
  - Receives an action a_t
  - Emits a state s_{t+1}
  - Emits a scalar reward r_{t+1}

Markov Decision Process



- The environment can be specified by:
  - A **state** space $\boldsymbol{S}$, $\boldsymbol{s} \in \boldsymbol{S}$ (discrete or continuous)
  - An **action** space $\boldsymbol{A}$, $\boldsymbol{a} \in \boldsymbol{A}$ (discrete or continuous)
  - **Transition probability distribution**
    - $P(s_{t+1}|s_t, a_t)$
    - Markov property
  - The initial state distribution ρ_0
  - **Reward function** $\mathrm{r}(s_t, a_t)$
  - The **discount factor** $\gamma \in [0,1]$
    - $\gamma \rightarrow 0$, consider only immediate reward

The objective of reinforcement learning problem

- Given a MDP, find the optimal policy π^* to maximize the expected return

$$J(\pi) = \mathbb{E}_{\tau \sim p_\pi(\tau)} \underbrace{\left[\sum_{t=0}^{\infty} \gamma^t r\left(\mathbf{s}_t, \mathbf{a}_t\right) \right]}_{\text{return } R(\tau)}$$

$$\pi^* = \arg\max_{\pi} J(\pi)$$

  - Trajectory (episode) : $\tau = (s_0, a_0, s_1, a_1, \ldots)$
  - Return $R(\tau)$: get bounded by γ
  - The probability of τ under π is

$$p_\pi(\tau) = \rho_0\left(\mathbf{s}_0\right) \prod_{t=0}^{\infty} \pi\left(\mathbf{a}_t \mid \mathbf{s}_t\right) P\left(\mathbf{s}_{t+1} \mid \mathbf{s}_t, \mathbf{a}_t\right)$$

Goal-oriented RL

- The agent aims to reach a set of **goal states** $\mathrm{G} = \{\mathrm{g}\}, \mathrm{G} \subseteq \mathrm{S}$ which indicating the success of a task
  - g is absorbing
- Stochastic shortest path problem via trial-and-error

$$r(s_t, a_t) = \begin{cases} R \geq 0, & reach\ goal\ state \\ r^c(s_t, a_t) < 0, & otherwise \end{cases}$$

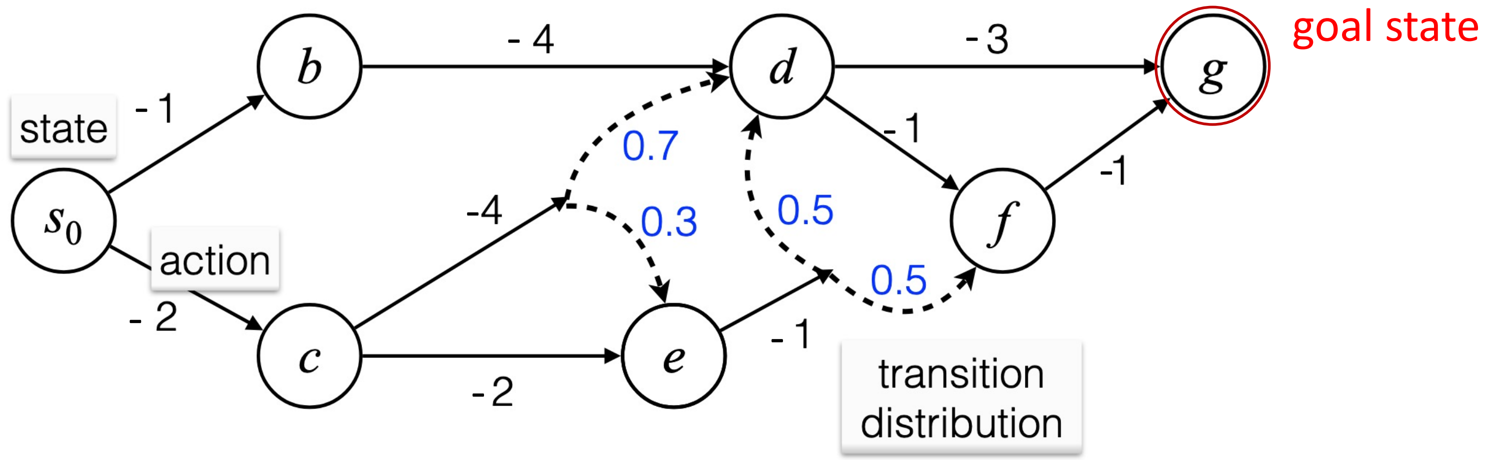
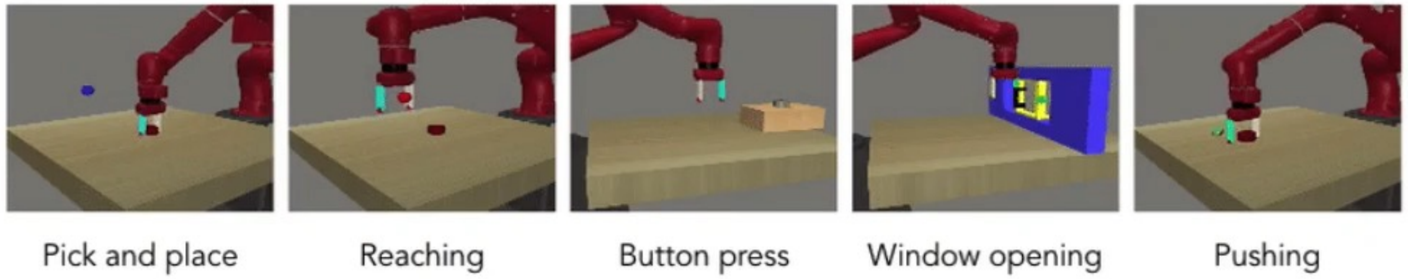


Image credit: Nan Jiang CS 498 Reinforcement Learning

Goal Matching Problem

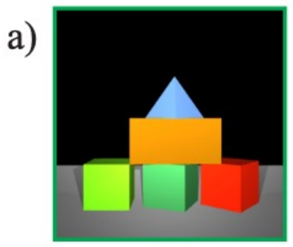
- Most of the robotics tasks can be formulated as goal-oriented RL problem, usually a single goal matching problem.



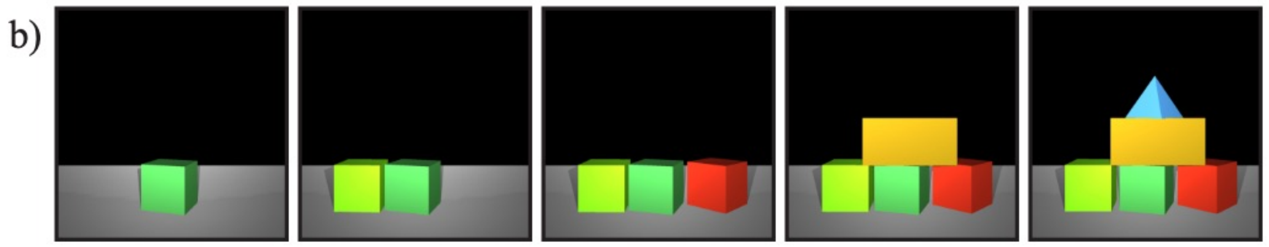
Goal state: target positions of the gripper and the objects

Yu, Quillen, He, et al., Meta-World: A Benchmark and Evaluation for Multi-Task and Meta Reinforcement Learning, 2019

Goal Matching Problem



goal image



build a block castle

goal image

visual navigation

Janner, Levine, Freeman, et al., Reasoning About Physical Interactions with Object-Oriented Prediction and Planning, 2019
Kolve, Mottaghi, Han, et al., AI2-THOR: An Interactive 3D Environment for Visual AI, 2019

Is goal matching enough for physics-awareness?

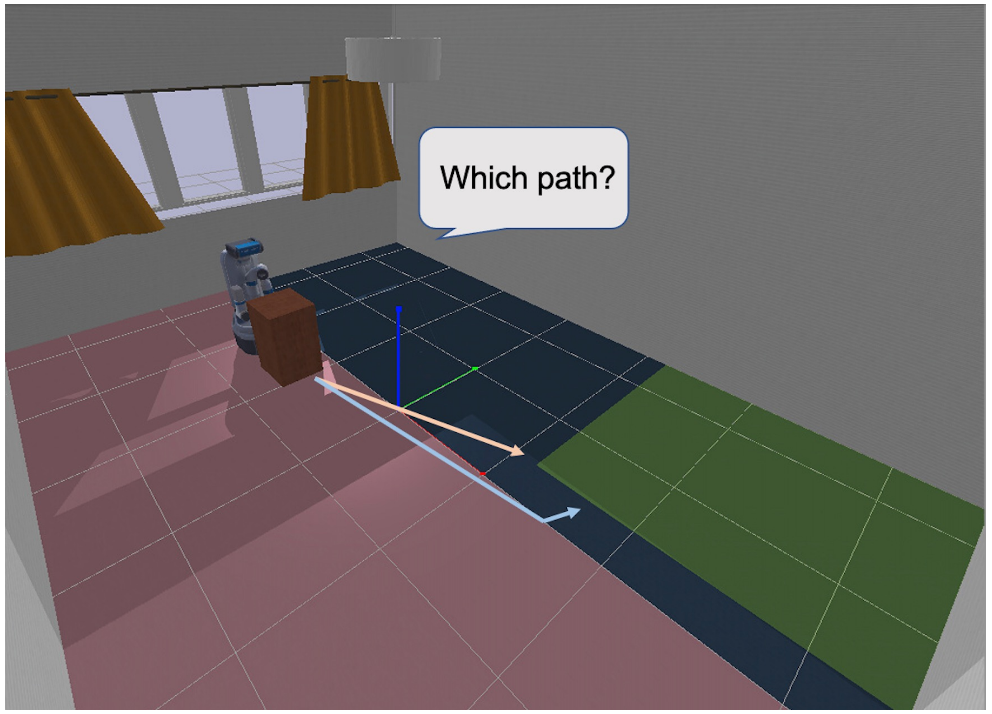
- Matching two goal states usually requires purely **geometric understanding** abilities
  - Match shapes, geometric configurations, latent visual representations and measure their distances
- Physics properties influence the agent's behavior in the mechanical process of how to reach the goal, not the goal itself.
  - Navigate through terrains with different frictions in different speed and gaze
  - Lift up a hammer of different weights to hit the nail requires different forces

How to evaluate an agent’s physics-awareness?

- The agent’s policy should reflect its understanding to the physical properties of the environment
  - The agent’s expected behavior should be consistent with human’s physics intuition
- The tasks should allow a set of equally valid goal states among which the agent must choose the optimal one based on its physical understanding

Variable Friction Pushing Task

- The agent should plan a trajectory to push a box to a target rectangular region located on the other band.
- Be aware of the friction difference



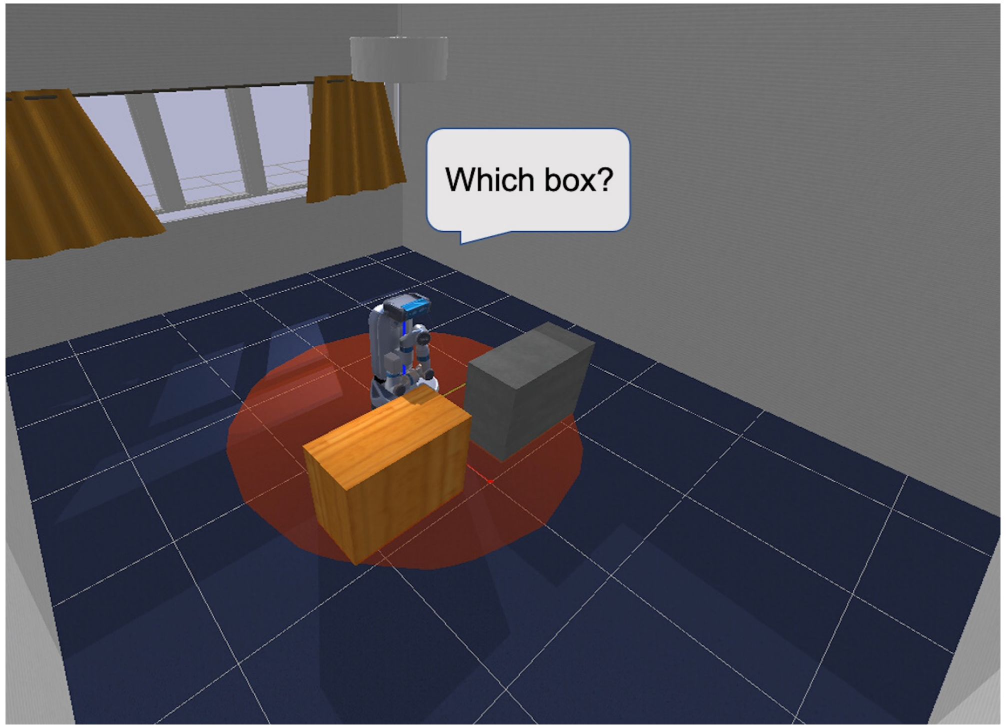
- The agent and a box are initialized on a two-band floor.
- The red and blue bands have friction coefficients 0.2 and 0.8 respectively.

Variable Mass Pushing Task

- The agent should push one of the boxes outside the circular area
- Be aware of the mass difference

  - The robot’s initial position is always at the center of the circle, while the two boxes are initialized at locations equidistant to the robot.

  - The two boxes are instances of the same CAD model but with different materials and weights (10 kg and 50 kg, respectively).



Energy Cost as Learning Signal

- To teach an agent the physics notions such as friction and mass in our pushing tasks, we compute the **raw energy** E(t) the agent spends at each time step t on pushing the box to move on a floor.
- E(t) is the virtual physical work done on translating and rotating the box

$$E(t) = E_{trans} + E_{rot}.$$

Adaptive Energy Cost

- The step energy cost

$$c(t) = \frac{E(t) - E_{min}(t)}{E_{max}(t) - E_{min}(t)}$$

  - Let $c(t) = 1$, when $E_{min}(t) = E_{max}(t)$.

- $E_{min}(t)$ and $E_{max}(t)$ are the running minimum and maximum of raw energy over the history until time step t.

$$E_{min}(t) = \min\{E(0), \cdots, E(t)\}$$

$$E_{max}(t) = \max\{E(0), \cdots, E(t)\}$$

Adaptive Energy Cost

- $E_{min}(t)$ and $E_{max}(t)$ caches the agent's current knowledge about the environment during the learning process
  - The physical properties of the visited locations
  - The agent's exploration behavior
  - How did the agent interact with the object
- $E_{min}(t)$ and $E_{max}(t)$ are updated in real time
  - Adding c(t) to the reward function yields a non-stationary reward function

Adaptive Energy Cost

- The normalized c(t) represents the per-step pushing energy cost relative to all history data

$$c(t) = \frac{E(t) - E_{min}(t)}{E_{max}(t) - E_{min}(t)}$$

- Then, we define $c(\tau)$ as the relative episode energy cost along a trajectory

$$c(\tau) = \sum_{t=0}^{H} c(t).$$

Variable Friction Pushing Task

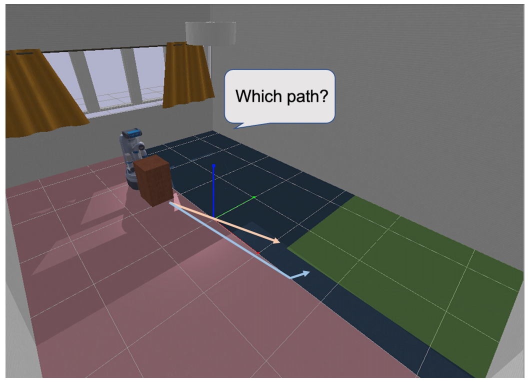
- State space
  - Current 3D positions and orientations of the robot and boxes in Cartesian space
- Action space
  - Robot platform: Fetch
    - Fixed torso, arm, hand
  - Velocity control on wheels
  - Discrete and parameterized by v which is the maximum value of wheel angular velocity
    - Move forward: [v, v]
    - Turn left: [0.5v, -0.5v]
    - Turn right: [−0.5v, 0.5v]
    - Stop: [0,0]

Variable Friction Pushing Task

- Basic setting
  - Reward function

$$r(\mathbf{s}_t, \mathbf{a}_t) = \begin{cases} 10 & \text{succeed} \\ -10 & \text{agent collides with obstacles (e.g. walls)} \\ -0.5 & \text{agent pushes box} \\ -1 & \text{otherwise} \end{cases}$$

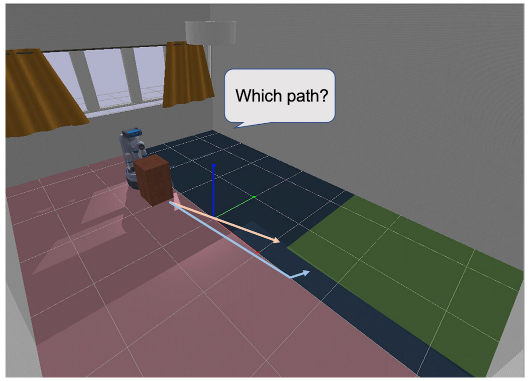
  - Optimal policy
    - Stochastic shortest path problem, penalize time elapse
    - Pushing the box to the target region as quickly as possible



Variable Friction Pushing Task

- Physics-aware setting
  - Reward function

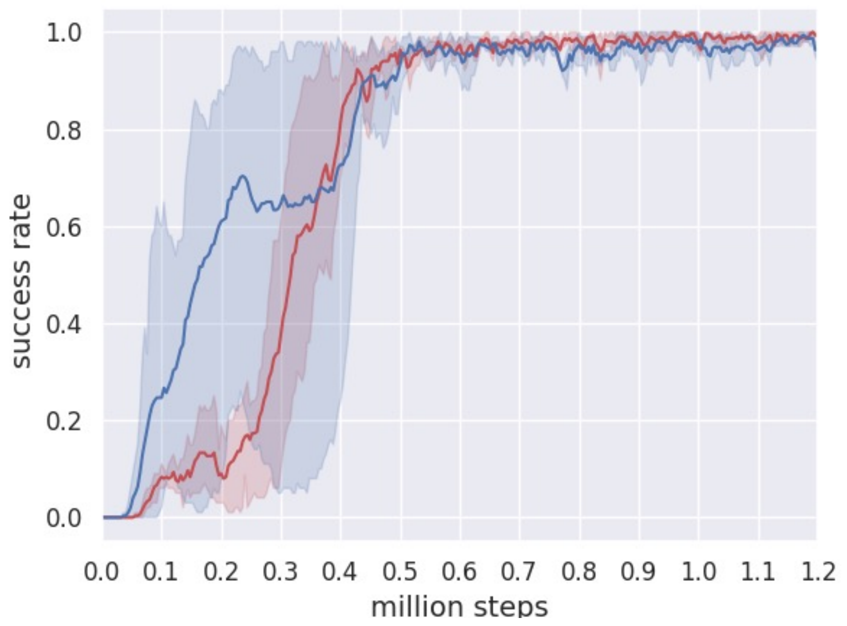
$$r(\mathbf{s}_t, \mathbf{a}_t) = \begin{cases} 10 & \text{succeed} \\ -10 & \text{agent collides with obstacles (e.g. walls)} \\ -0.5 \cdot \boxed{c(t)} & \text{agent pushes box} \\ -1 & \text{otherwise} \end{cases}$$



  - Optimal policy
    - Stochastic shortest path problem, penalize per step energy cost
      - Edges on MDP are pushed up and down according to the energy ratio in real-time
        - This landscape shaping helps the agent to memorize the optimal trajectory
    - Pushing the box to the target region in the most energy efficient way
      - Prefer to choose the path staying on the low-friction band longer

Variable Friction Pushing Task: Training Curves

- Are the proposed energy-aware reward functions help learn an energy-efficient policy?



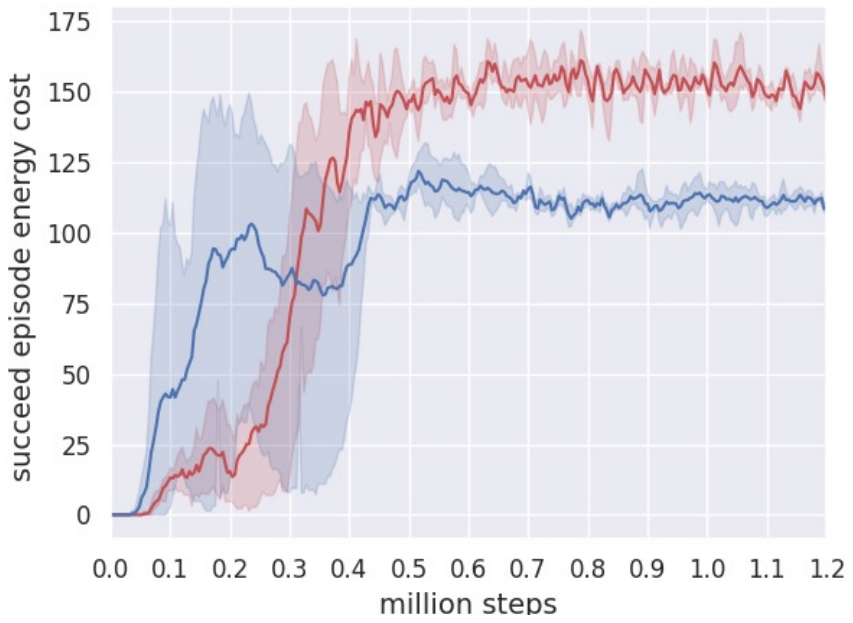
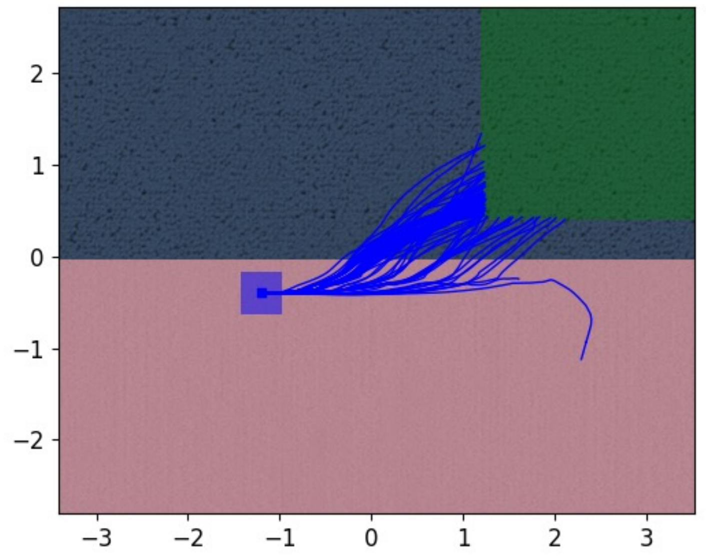


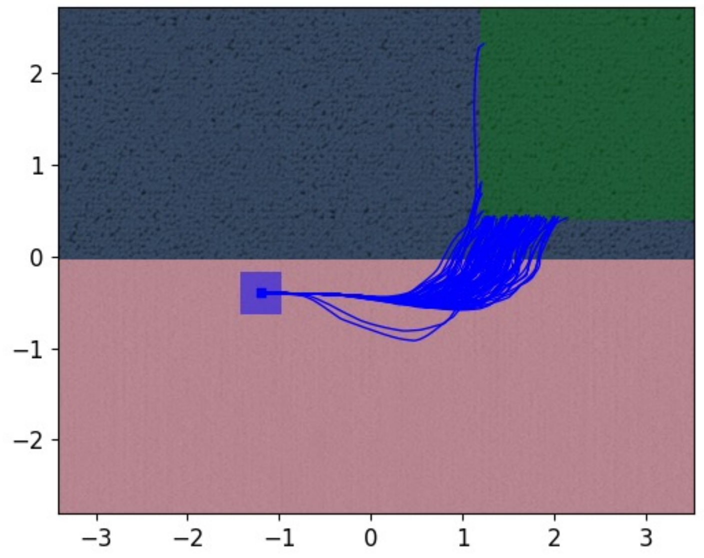
Figure 3: Training curves of PPO on variable friction pushing tasks. The algorithm is run across three different random seeds. Left: successful episode energy cost. Right: success rate. Red: baseline policy trained without energy rewards. Blue: energy-aware policy trained with energy rewards.

Variable Friction Pushing Task: Trajectory Distribution

- Does the behavior difference between policies learned with or without energy rewards reflect the agent’s knowledge of friction coefficient?



No energy: prefer the shortest path

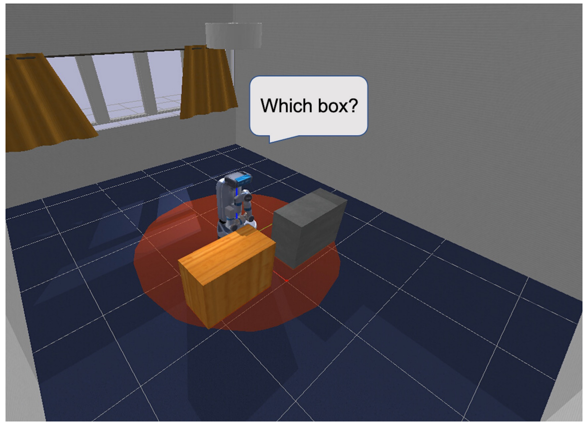


With energy: prefer the low friction path

Variable Mass Pushing Task

- Basic setting
  - Reward function

$$r(\mathbf{s}_t, \mathbf{a}_t) = \begin{cases} 100 & \text{succeed} \\ -10 & \text{agent collides with obstacles (e.g. walls)} \\ 0 & \text{otherwise} \end{cases}$$



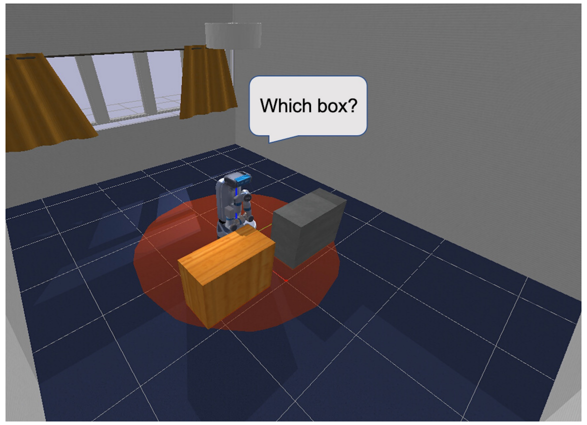
  - Optimal policy
    - Choose either box to push outside the circular region
    - Not a stochastic shortest path problem
      - The pushing trajectory does not matter
      - Choose which box to push is expected to have no correlation with the box weight.

Variable Mass Pushing Task

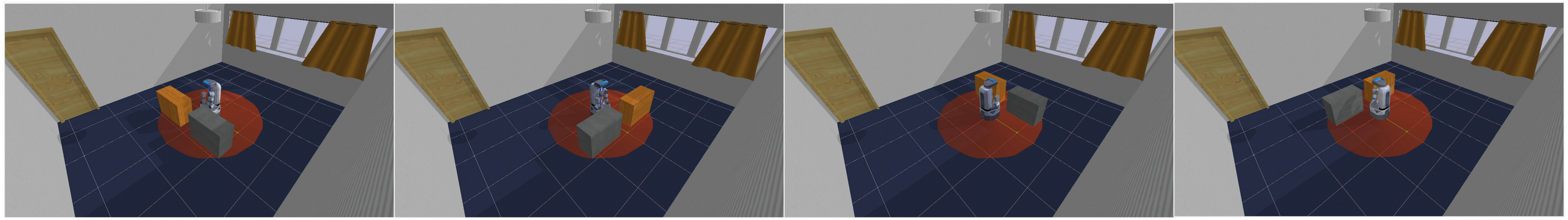
- Physics-aware setting
  - Reward function

$$r(\mathbf{s}_t, \mathbf{a}_t) = \begin{cases} 100 \cdot \boxed{c(\tau)} & \text{succeed} \\ -10 & \text{agent collides with obstacles (e.g. walls)} \\ 0 & \text{otherwise} \end{cases}$$

  - Optimal policy
    - Always choose to push the light box outside the circular region
    - Trajectories on MDP are pushed up and down according to the energy ratio in real-time
      - This landscape shaping helps the agent to memorize the optimal trajectory



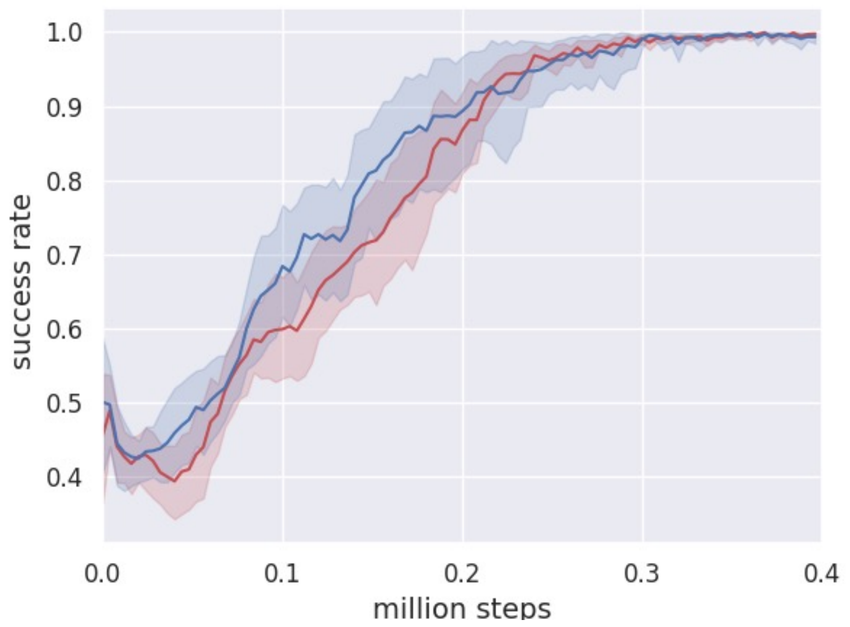
Variable Mass Pushing Task: Training Configurations


Configuration 0

- In each configuration
  - We allow for four facing directions as the agent's initial orientation to diversify the initial geometric configuration.
  - For each facing direction, there are two choices to place the box: the light one on the left or on the right.
- Each training trial will correspond to a configuration covering all four directions.
  - 16 configurations in total.
  - At the beginning of every episode, one facing direction is randomly selected from the certain configuration.
- Train on randomly picked 10 configurations to prevent the agent from memorizing the correlations between box positions and masses.

Variable Mass Pushing Task: Training Curves

- Are the proposed energy-aware reward functions help learn an energy-efficient policy?



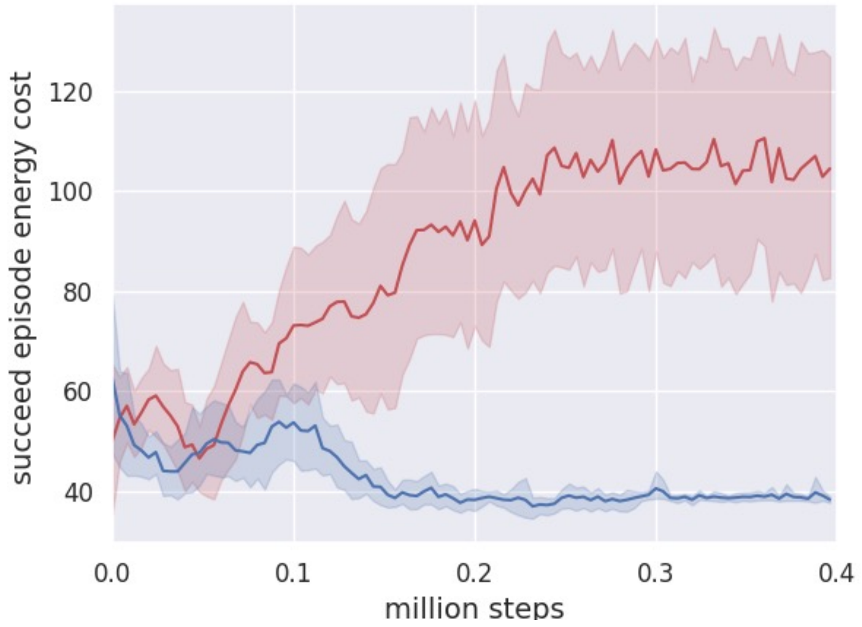


Figure 5: Training curves of PPO on variable mass pushing tasks, run across 10 configurations randomly selected out of 16. Left: successful episode energy cost. Right: success rate. Red: baseline policy trained without energy rewards. Blue: energy-aware policy trained with energy rewards.

Variable Mass Pushing Task: Trajectory Distribution

- Does the behavior difference between policies learned with or without energy rewards reflect the agent's knowledge of mass?

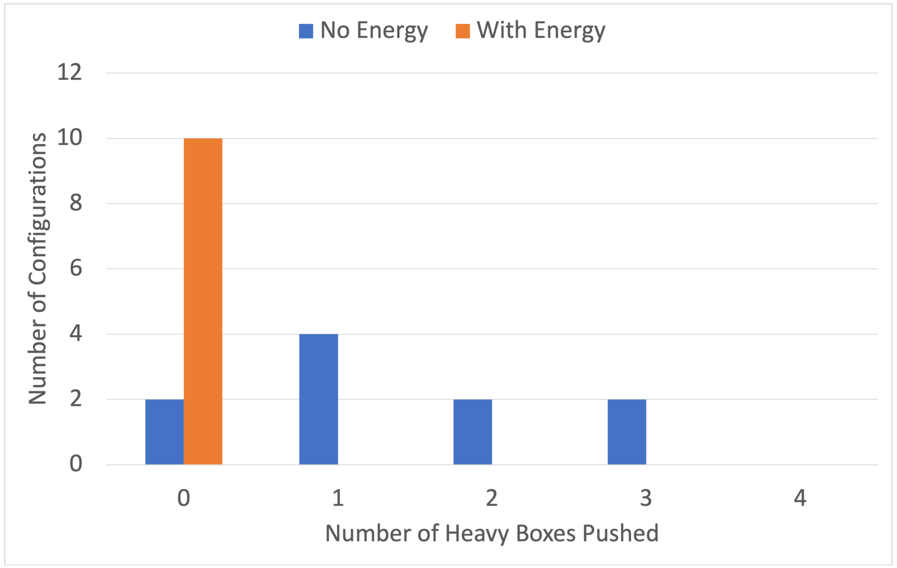


Figure 7: Frequency distributions of the number of heavy boxes pushed over 10 configurations. Blue: baseline agent without energy. Orange: energy-aware agent.

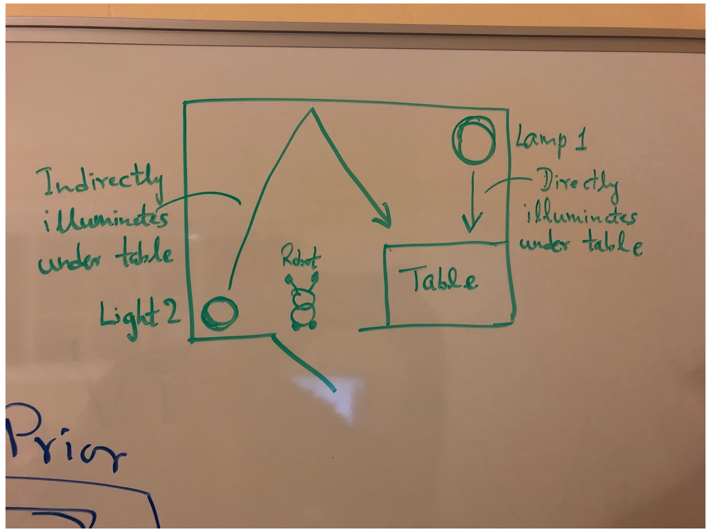
- The energy-aware agent always pushes the lighter box in all 10 configurations.
- The baseline agent chooses to push the light or heavy box with nearly equal probabilities.

Next problem 1: Lighting-aware Agents

- Motivation:
  - Can the robot be aware of lighting and act accordingly?
- Task scenario 1
  - Extension of the variable friction pushing task with images as observations
  - The agent is given the images of two materials before the pushing task and needs to choose a path with low energy cost according to this prior knowledge.

Next problem 1: Lighting-aware Agents

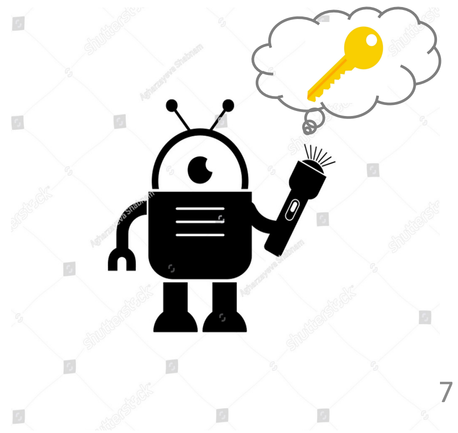
- Task scenario 2
  - This task is designed to test the robot's understanding ability of global illumination.
  - A robot is asked to fetch the key from under a dark table. The room has two lights. Turing on either light will give enough illumination for the robot to see the key, but turning on the right light and pick up the key will take longer path.
  - The robot is aiming to succeed the task in minimum amount of time.



Lamp 1
Directly illuminates under table
Indirectly illuminates under table
Robot
Table
Light 2
Prior

Next problem 1: Lighting-aware Agents

- Task scenario 3
  - An agent is equipped with a flashlight and aiming to search for a given object in a dark room in minimum amount of time
    - Similar to a robot rescue task
  - The target object is given as an image
  - The agent could navigate in the 3D room and decide when to turn on the flashlight and where to look at.



Next problem 1: Lighting-aware Agents

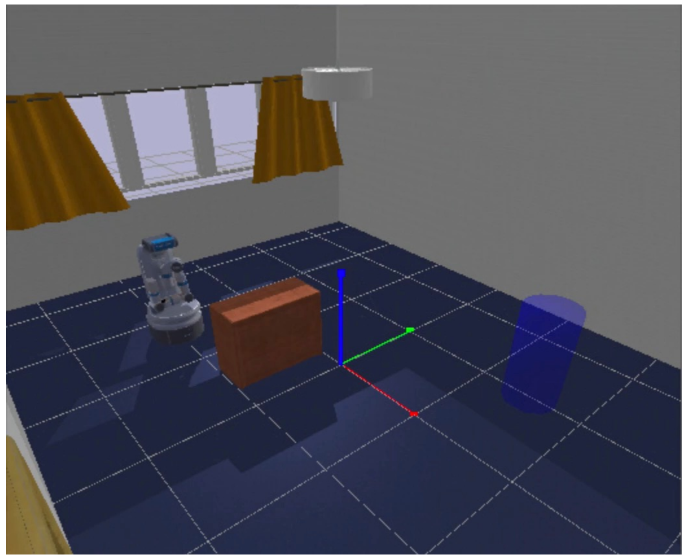
- Active exploration
  - Explore the space
    - Never seen before
    - The model uncertainty is high
    - More likely target locations inferred from prior knowledge
      - high correlation between pillow and bed
- Build the environment model
  - Embed the first-person visual observations into a low dimensional latent state space z
    - Preserve the visual similarity for retrieving the image of target object
  - **Key challenge: Build the MDP graph in the latent space Z**
    - **Preserve the transitions:** $s, a \rightarrow s'$
    - **Preserve the spatial relationships of the objects** (The cup is on the table)

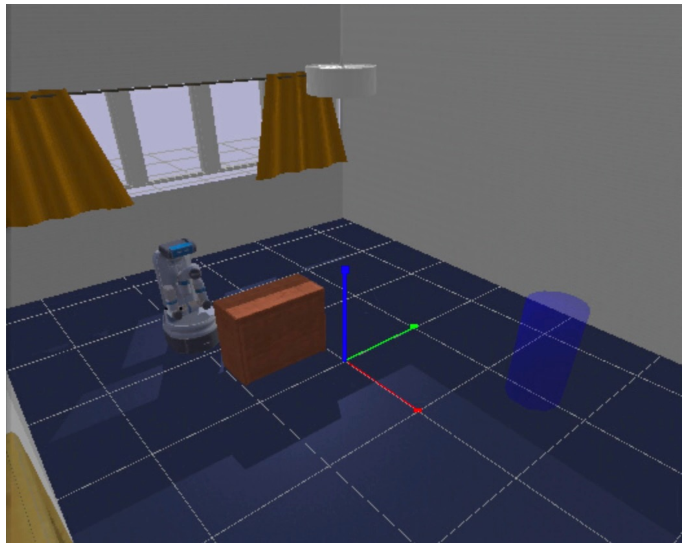
Next problem 1: Lighting-aware Agent

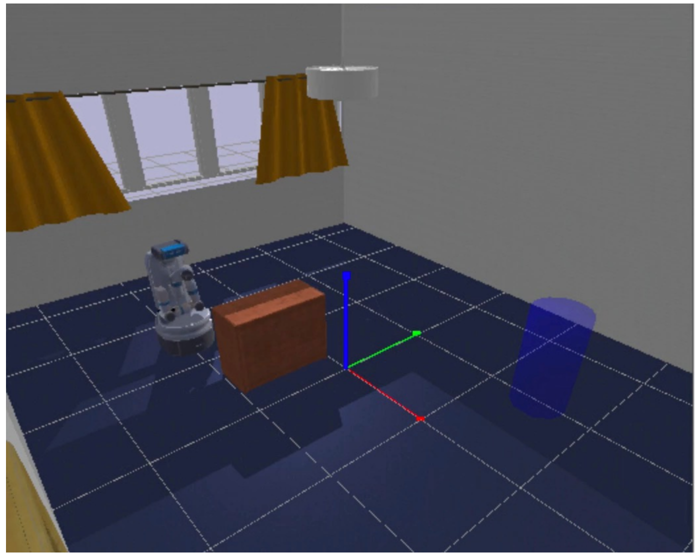
- Timeline
  - Create the environment and tasks, show toy examples
    - 2~3 months (may need better rendering, adapt robot morphology)
  - Study the key problems
    - 4 months (vision-based robot learning requires longer training time on large computational resources)
  - Expected publication
    - CVPR 2022
      - Submission deadline: Nov 2021
    - IROS 2022
      - Submission deadline: Mar 2022
    - ECCV 2022
      - Submission deadline: Mar 2022

Next problem 2: Transfer skills to different dynamics?

- **Motivation**: If the physics parameter of the environment changes, can the agent ***be aware of*** that and adapt its policy accordingly?
- Example: Train: mass=10, friction coefficient=0.8

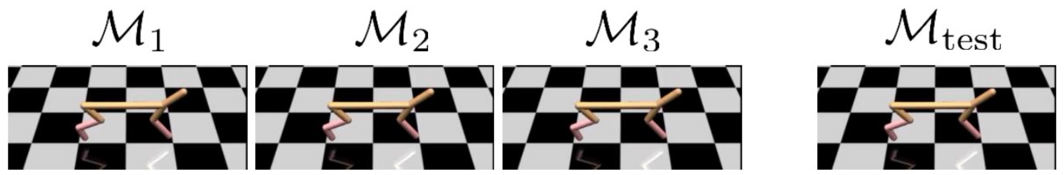

Test: mass=10, friction coefficient=0.8


Test: mass=10, **friction coefficient=0.1**


Test: **mass=150**, friction coefficient=0.8

Transfer learning in RL: Problem Formulation

- Assumption: $M_i \sim P(\mathcal{M})$
  - In our case, training and test MDPs come from the same distribution with changes in dynamics $P(s_{t+1}|s_t, a_t)$
- Training
  - Train on MDPs $\{M_1, \dots, M_n\}$, find $\pi_1, \dots, \pi_n$



friction=0.1 friction=0.4 friction=0.6 friction=0.8

- Test-time
  - Sample a new MDP $M_{test} \sim P(\mathcal{M})$
  - Find π_{test} leveraging **experiences and knowledge** from $\{M_1, \dots, M_n\}$ for faster learning and better performance (i.e. how to act and explore)
    - Trajectories from $\{M_1, \dots, M_n\}$
    - Policies and value functions
    - Transition functions

Adapted from Sergey Levine, CS 285 Deep Reinforcement Learning

Meta Learning Perspective

- Key idea: Policies π_{ϕ_i} are generated from a higher-level function f_θ given the MDP
- Training:

$$\theta^\star = \arg\max_\theta \sum_{i=1}^{n} E_{\pi_{\phi_i}(\tau)}[R(\tau)]$$

$$\text{where } \phi_i = f_\theta(\mathcal{M}_i)$$

$$\begin{tikzcd} M_i \arrow[r] & \boxed{f_\theta} \arrow[r] & \phi_i \end{tikzcd}$$

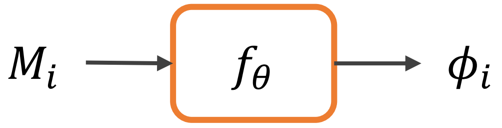
- Test-time:
  - Given M_{test}, find π_{test} from ϕ_{test}=$f_{\theta^*}(M_{test})$

Meta Learning Perspective

- In Practice, three ways to implement f_θ :
  - f_θ is a RNN policy trained across MDPs
  - Bi-level optimization: learn π_θ

$$f_\theta(\mathcal{M}_i) = \theta + \alpha \nabla_\theta J_i(\theta)$$

  - Inference problem
    - The dynamics are generated from a distribution parameterized by z
      - In our case, z is physics parameters such as friction, mass.
    - Find policy $\pi_\theta(a|s,z)$ conditioning on z



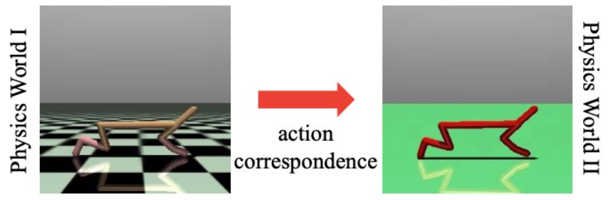
$$\begin{tikzcd} M_i \arrow[r] & \boxed{f_\theta} \arrow[r] & \phi_i \end{tikzcd}$$

Dynamics Alignment Perspective

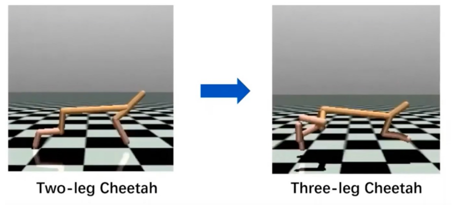
- We cannot transfer because we are not abstract enough
  - Align MDPs to abstract out dynamics differences in friction, mass, …
- Find a mapping H from M_1 to M_2 such that

$$\begin{tikzcd}
s_t^1 \arrow[r, "a_t^1"] \arrow[d, dashed, "H"] & s_{t+1}^1 \arrow[d, dashed, "H"] & M_1 \\
s_t^2 \arrow[r, "{a_t^2, \dots, a_{t+k-1}^2}"] & s_{t+k}^2 & M_2
\end{tikzcd}$$

- Once this mapping is found, we can directly transfer the policy trained on M_1 to M_2, without needing any additional learning on M_2.



(b) HalfCheetah cross physics alignment



Transfer two-leg cheetah policy to three-leg cheetah

Zhang, Xiao, Efros, et al., Learning Cross-Domain Correspondence for Control with Dynamics Cycle-Consistency, 2020

Next problem 2: Transfer learning in RL across different dynamics

- Timeline
  - Study and improve the existing algorithms on RL benchmark tasks
    - 3 months
  - Propose new algorithm
    - 3 months
  - (Experiments on complex real-world tasks: 2 months)
  - Expected publication
    - ICLR 2022
      - Submission deadline: Sept 2021
    - Neurips 2022
      - Submission deadline: May 2022
    - ICLR 2023
      - Submission deadline: Sept 2022

Q & A

Thanks!

Backup: PPO

- Policy gradient algorithm
- Optimization objective

$$J(\theta') - J(\theta) = E_{\tau \sim p_{\theta'}(\tau)}\left[\sum \gamma^t A^{\pi_\theta}(\mathbf{s}_t, \mathbf{a}_t)\right]$$

$$J(\pi) = \mathbb{E}_{\tau \sim p_\pi(\tau)}\left[\sum_{t=0}^{\infty} \gamma^t r(\mathbf{s}_t, \mathbf{a}_t)\right]$$

- Approximated objective

$$\theta' \leftarrow \arg\max_{\theta'} \sum_t E_{\mathbf{s}_t \sim p_\theta(\mathbf{s}_t)}\left[E_{\mathbf{a}_t \sim \pi_\theta(\mathbf{a}_t|\mathbf{s}_t)}\left[\frac{\pi_{\theta'}(\mathbf{a}_t|\mathbf{s}_t)}{\pi_\theta(\mathbf{a}_t|\mathbf{s}_t)} \gamma^t A^{\pi_\theta}(\mathbf{s}_t, \mathbf{a}_t)\right]\right]$$

$$\text{such that } D_{\text{KL}}(\pi_{\theta'}(\mathbf{a}_t|\mathbf{s}_t) \| \pi_\theta(\mathbf{a}_t|\mathbf{s}_t)) \leq \epsilon$$

for small enough ϵ, this is guaranteed to improve $J(\theta') - J(\theta)$

Backup: PPO

- Adaptive KL PPO:
  - Solve the constrained optimization problem by adaptive KL penalty coefficient

$$L^{KLPEN}(\theta) = \hat{\mathbb{E}}_t\left[\frac{\pi_\theta(a_t \mid s_t)}{\pi_{\theta_{\text{old}}}(a_t \mid s_t)}\hat{A}_t - \beta \, \text{KL}[\pi_{\theta_{\text{old}}}(\cdot \mid s_t), \pi_\theta(\cdot \mid s_t)]\right]$$

- Compute $d = \hat{\mathbb{E}}_t[\text{KL}[\pi_{\theta_{\text{old}}}(\cdot \mid s_t), \pi_\theta(\cdot \mid s_t)]]$
  - If $d < d_{\text{targ}}/1.5$, $\beta \leftarrow \beta/2$
  - If $d > d_{\text{targ}} \times 1.5$, $\beta \leftarrow \beta \times 2$

Backup: PPO

- Clip PPO:
  - Surrogate objective
    - Approximate the KL constraint by clipping the advantage function

$$L^{CPI}(\theta) = \hat{\mathbb{E}}_t\left[\frac{\pi_\theta(a_t \mid s_t)}{\pi_{\theta_{\text{old}}}(a_t \mid s_t)}\hat{A}_t\right] = \hat{\mathbb{E}}_t\left[r_t(\theta)\hat{A}_t\right]$$

$$L^{CLIP}(\theta) = \hat{\mathbb{E}}_t\left[\min(r_t(\theta)\hat{A}_t, \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t)\right]$$

Backup: PPO

- Add entropy term to the objective to encourage exploration

$$\underset{\theta}{\text{maximize}} \qquad \mathbb{E}_{\pi_{\theta_k}} \sum_{t=0}^{\infty} f(r(\theta; s_t, a_t), A(s_t, a_t)) + \eta \cdot \text{ entropy}(\pi_{\theta'}(s_t, \cdot))$$

Backup: Transfer Learning in RL

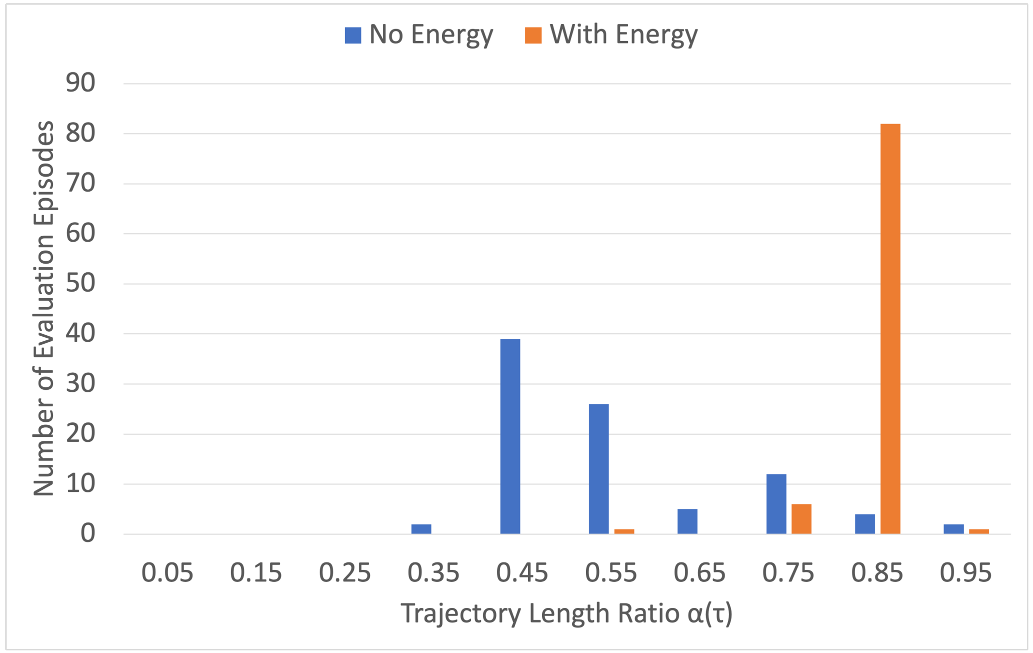
- Finetuning does not work
  - Policies and value functions become overly specialized
  - Optimal policies trained in fully observed MDPs are deterministic
    - Wil not explore in the new MDP
    - Low-entropy policies adapt very slowly to new settings

Backup: Object searching in the dark

- Active exploration
  - Turing on or off the light will sharply change the visual fields (observations)
  - Explore the space
    - Never seen before
    - The model uncertainty is high
    - More likely locations inferred from prior knowledge (high correlation between pillow and bed)
- Model building
  - Embed the first-person visual observations into a low dimensional latent state space z
    - Preserve the visual similarity for retrieving the image of target object
  - **Key challenge: Build the MDP graph in the latent space Z**
    - **Preserve the transitions:** $s, a \rightarrow s'$
    - **Preserve the spatial relationships of the objects** (The cup is on the table)
      - **Searching with the flashlight results in a limited visual field which makes the inference of object spatial relationship harder**

Variable Friction Pushing Task: Trajectory Distribution

- Does the behavior difference between policies learned with or without energy rewards reflect the agent's knowledge of friction coefficient?

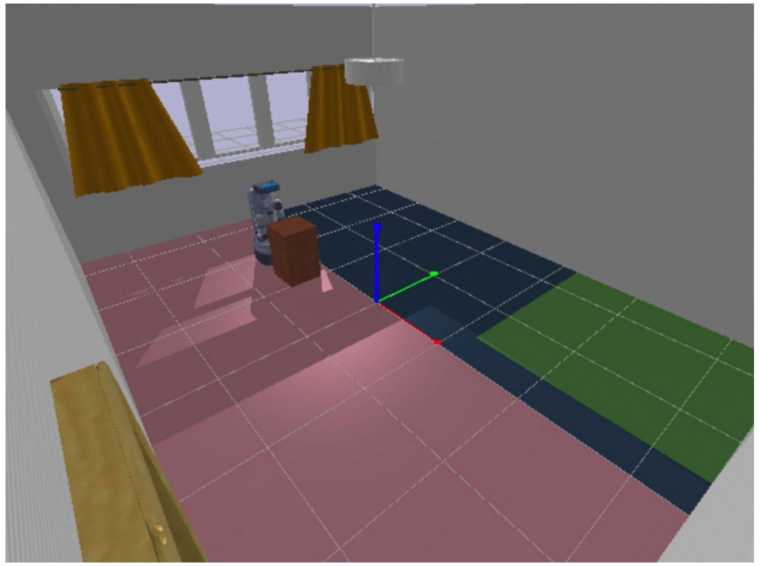


$$\alpha(\tau) = \frac{l(\tau_{low})}{l(\tau)}$$

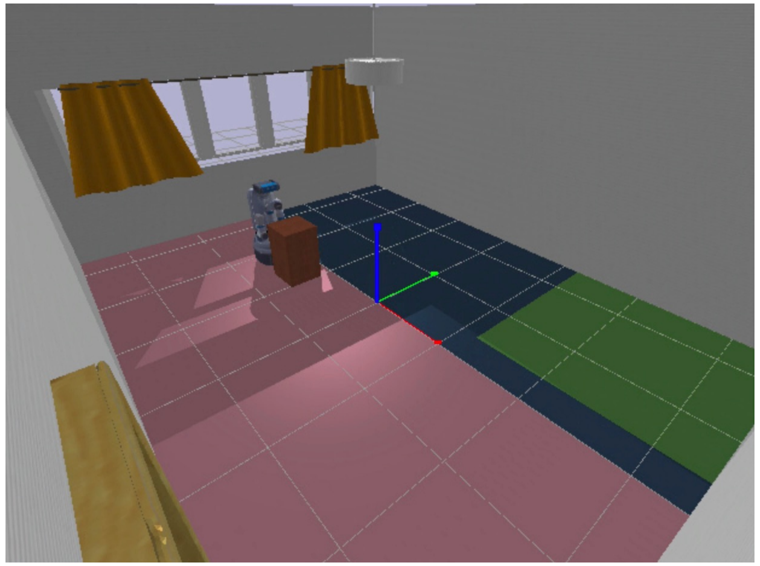
The fraction of trajectory length spent on the low friction band

Figure 6: Frequency distributions of trajectory length ratio $\alpha(\tau)$ of evaluation trajectories. Blue: baseline agent without energy. Orange: energy-aware agent.

Variable Friction Pushing Task

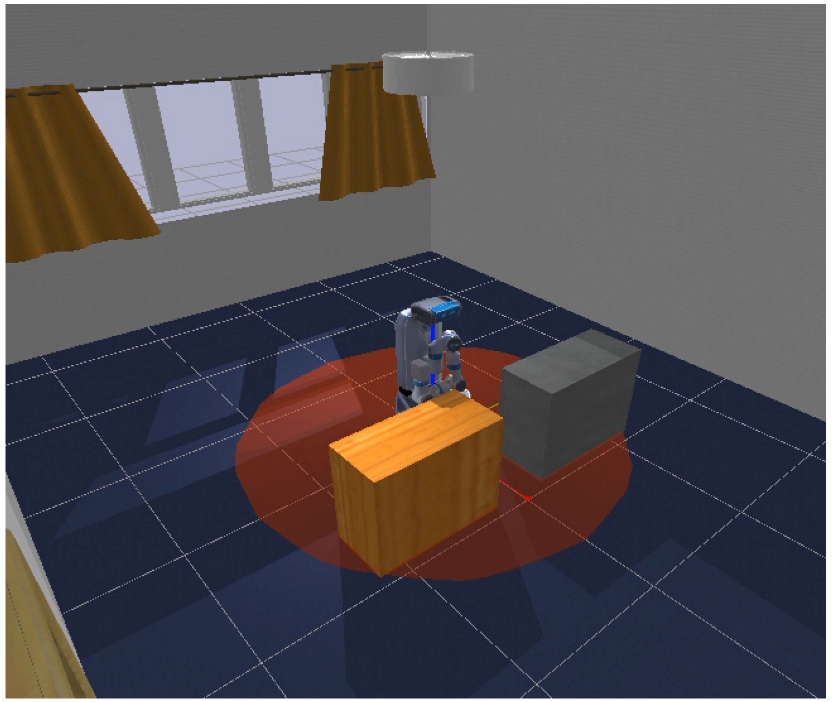


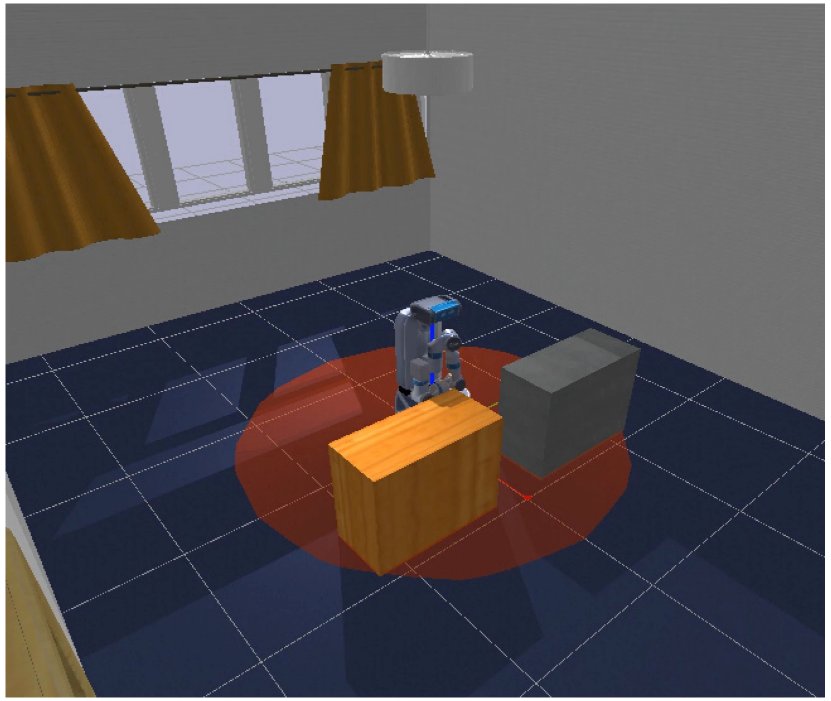
No energy: prefer the shortest path



With energy: prefer the low friction path

Variable Mass Pushing Task


No energy: choose either box


With energy: always choose lighter box