Lecture 11
Markovian Modeling II

AncoraSIR.com

[Please refer to the course website for copyright credits]

SUSTech
Southern University of Science and Technology

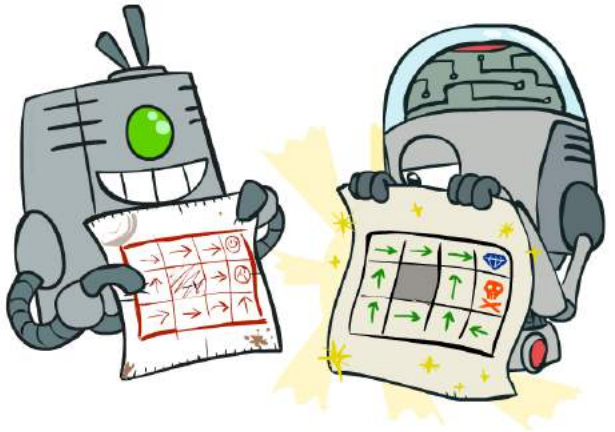
Policy Iteration

AncoraSIR.com

SUSTech
Southern University of Science and Technology

The Policy Iteration Algorithm

If one action is clearly better than all others,
then the exact magnitude of the utilities on the states involved need not be precise



- **Starts**
  - with some initial policy π_0
- **Repeat**
  - Policy evaluation
    - Given a policy π_i, calculate $U_i = U^{\pi_i}$, the utility of each state if π_i were to be executed.
  - Policy improvement
    - Calculate a new MEU policy π_{i+1}, using one-step look-ahead based on U_i
- **Terminates**
  - when the policy improvement step yields no change in the utilities

AncoraSIR.com

The Policy Iteration Algorithm

for calculating an optimal policy

function POLICY-ITERATION(mdp) **returns** a policy
  inputs: mdp, an MDP with states S, actions $A(s)$, transition model $P(s' \mid s, a)$
  local variables: U, a vector of utilities for states in S, initially zero

Start → π, a policy vector indexed by state, initially random

Repeat

repeat

- Evaluation → $U \leftarrow$ POLICY-EVALUATION(π, U, mdp)

  $unchanged? \leftarrow$ true

- Improvement

  for each state s **in** S **do**

    if $\max_{a \in A(s)} \sum_{s'} P(s' \mid s, a)\, U[s'] > \sum_{s'} P(s' \mid s, \pi[s])\, U[s']$ **then do**

      $\pi[s] \leftarrow \operatorname{argmax}_{a \in A(s)} \sum_{s'} P(s' \mid s, a)\, U[s']$

      $unchanged? \leftarrow$ false

Terminate → **until** $unchanged?$

return π

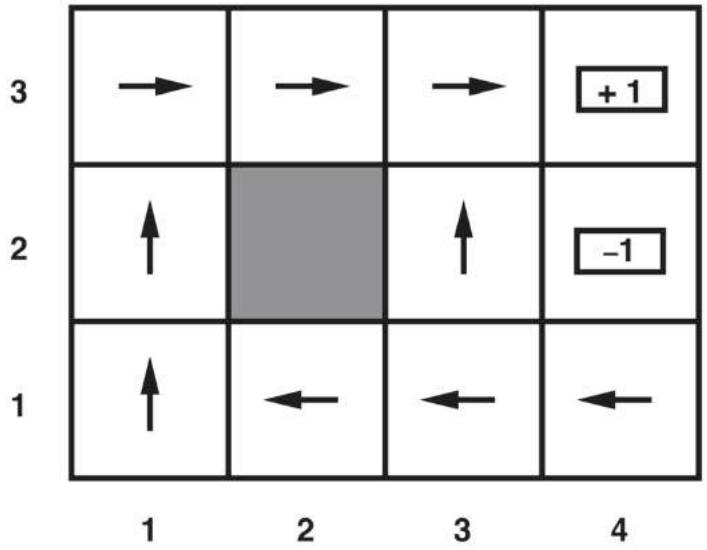
AncoraSIR.com

A Simplified Version of the Bellman Equation

the action in each state is fixed by the policy

- At the ith iteration, the policy π_i specifies the action $\pi_i(s)$ in state s

$$U(s) = R(s) + \gamma \max_{a \in A(s)} \sum_{s'} P(s' \mid s, a)\, U(s') \longrightarrow U_i(s) = R(s) + \gamma \sum_{s'} P\big(s' \mid s, \pi_i(s)\big)\, U_i(s')$$



	n states	n equations	n unknowns
π_i	$\pi_i(1,1) = Up$	$U_i(1,1) = -0.04 + 0.8U_i(1,2) + 0.1U_i(1,1) + 0.1U_i(2,1)$	
	$\pi_i(1,2) = Up$	$U_i(1,2) = -0.04 + 0.8U_i(1,3) + 0.2U_i(1,2)$	
	$\pi_i(1,3) = Right$	$U_i(1,3) = ?$	
	$\pi_i(2,1) = Left$	...	
	...	...	

Linear Simplified Bellman Equations

- Can be solved in time $O(n^3)$
- Efficient for small state spaces
- What about large state spaces?

The Simplified Bellman Process

it is not necessary to do exact policy evaluation

- Perform some number of simplified value iteration steps to give a reasonably good approximation of the utilities

$$U_i(s) = R(s) + \gamma \sum_{s'} P\big(s' \mid s, \pi_i(s)\big)\, U_i(s') \xrightarrow{\text{Repeat } k \text{ times}} U_{i+1}(s) \leftarrow R(s) + \gamma \sum_{s'} P\big(s' \mid s, \pi_i(s)\big)\, U_i(s')$$

- Modified Policy Iteration
  - Much more efficient than standard policy iteration or value iteration
- Asynchronous Policy Iteration
  - It is not necessary to update the utility or policy for all states at once
  - On each iteration, we can pick *any subset* of states and apply *either* kind of updating (policy improvement or simplified value iteration) to that subset

AncoraSIR.com

Formalizing Manipulation Learning

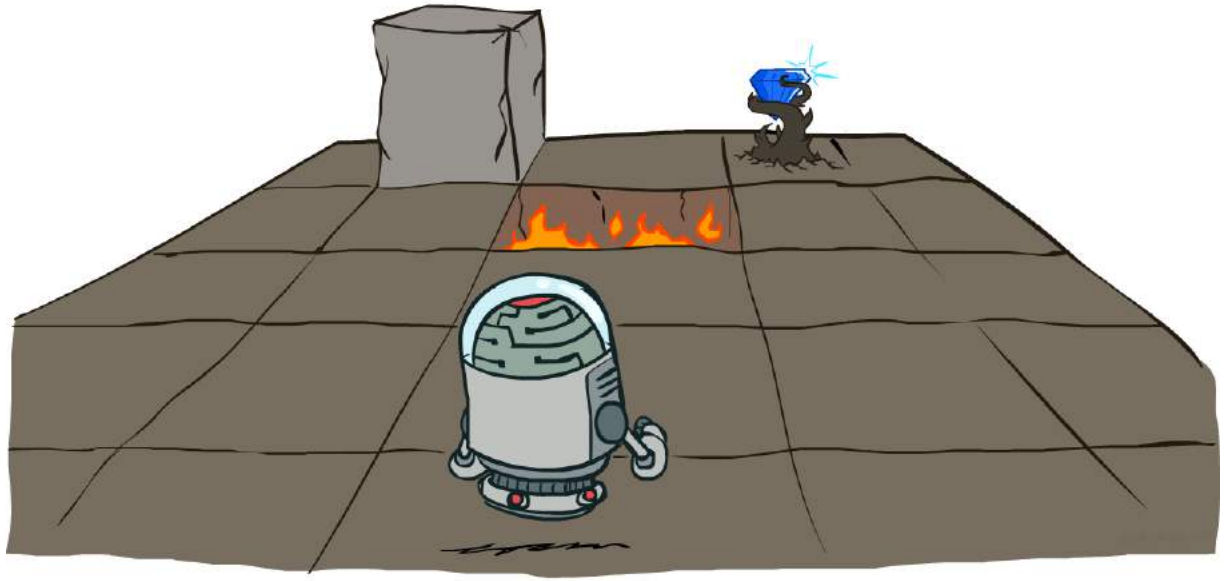
AncoraSIR.com

SUSTech
Southern University of Science and Technology

Formalizing Robot Learning Problems

Typically formulated as individual Markov Decision Processes (or MDPs)

- A 5-tuple formalization (S, A, T, R, γ)
  - A finite set of **states** $S \subseteq \mathbb{R}^n$
  - A finite set of **actions** $A \subseteq \mathbb{R}^m$
  - A **transition function** $T(s' | s, a)$
  - A **reward function** $R(s' | s, a)$
  - A **discount factor** $\gamma \in [0, 1]$



- Goal of Learning
  - To find a control policy $\pi_s^* = \text{argmax}_\pi \sum_{t=0}^{\infty} \gamma^t R(s_t)$

To identify and exploit task structure to obtain faster learning and better generalization

- POMDPs (Partially Observable MDPs)
  - A more accurate characterization, but more difficult to solve

AncoraSIR.com

Structured Collection of MDPs as a Task Family

Ideally attempt to learn a policy that generalizes across the entire task family

- Task Family
  - A distribution, $P(M)$, over MDPs, each of which is a *task*.
- Action Space
  - Determined by the robot and remains the same across tasks
  - Each task may have its own state space and transition and reward functions
- Reward Function
  - Typically formulated as a robot-dependent background cost function
  - Shared across the entire family
  - Plus a terminal reward that is specific to that task

$$R_i = G_i - C$$

the reward function for the i^{th} task

the general background cost function

AncoraSIR.com

Structured Collection of MDPs as a Task Family

Ideally attempt to learn a policy that generalizes across the entire task family

- State Space
  - The state space of the i^{th} task

$$S_i = S_r \times S_{e_i}$$

the state of the robot

the state of the i^{th} environment

- raw pixels and sensor values
- a highly pre-processed collection of relevant task variables

- Factorize the environment as a collection of object states
  - Many task environments will consist of ***a collection of objects*** and the variables describing ***the aspects of those objects*** that are relevant to the task

$$S_{e_i} = S_{\omega_i} \times \Omega_1^i \times \ldots \times \Omega_{k_i}^i$$

the state of the general environment

the state of the j^{th} relevant object in task i, and task i contains k_i objects

- Facilitates **object-centric** generalization
  - Can be **reused** in new environments containing the same objects

AncoraSIR.com

Structure in Transition Function

the robot will only be able to affect a subset of objects and state variables from any given state

- Modularity in Transition Functions
  - Model the tasks as hybrid systems with piecewise continuous dynamics

- **Mode**: *Each of the continuous dynamical subsystems*
  - the state will often contain discrete variables to capture the current mode

- **Mode Switches**: *occur when the robot enters certain sets of states*
  - when the robot makes or breaks contact with an object.

- Limit the state variables that the robot may alter by restricting itself to certain modes

AncoraSIR.com

Structure in Action Space, or *Skills*

Exploited using higher-level actions

- Typically modeled using a hierarchical learning framework
  - Models each motor skill as an *option*: $o = (I_o, \beta_o, \pi_o)$
  - I_o: $S \rightarrow \{0, 1\}$ is the *initiation set*
    - an indicator function describing the states from which the option may be executed.
  - β_o: $S \rightarrow [0, 1]$ is the *termination condition*
    - describing the probability that an option ceases execution upon reaching state s.
  - π_o is the *option policy*
    - mapping states in the option's initiation set to low-level motor actions and corresponding to the motor skill controller.
- Examples of the robot action space
  - Pre-equipped with a set of motor skills to reuse across the family of tasks
  - Discovers reusable skills as part of its learning task

AncoraSIR.com

SUSTech
Southern University of Science and Technology

A Dilemma between a Single Task and a Task Family

Challenges when transfer learned functions across a task family

- Learned policies for a single task may not share the same task space with others
  - Different tasks in a task family may require different functions of the task state

- Adding extra information to the task
  - for example, information about the color and shape of various objects in the task
  - Incompatible with the state space representation (no change)

- Extra information modelled as a *context vector* τ
  - Added to each task MDP, and which the robot can use to inform its behavior.
  - Can be monolithic for each task, or factored into object contexts.

AncoraSIR.com

Modeling a Family of Manipulation Tasks

a task family specified by a distribution of manipulation MDPs, $P(M)$

- A 6-tuple $M_i = (S_i, A, R_i, T_i, \gamma, \tau_i)$ of $P(M)$
  - $S_i = S_r \times S_{e_i}$ is the **state space**
    - Where S_r describes the robot state, and S_{e_i} the environment state. Often the environment state is primarily factored into a collection of object states: $S_{e_i} = S_{\omega_i} \times \Omega_1^i \times \ldots \times \Omega_{k_i}^i$, for a task with k_i objects;
  - A is the **action space**
    - Common across tasks, which may include both low-level primitive actions and a collection of options O;
  - $R_i = G_i - C$ is the **reward function**
    - Comprising a background cost function C (common to the family) and a task-specific goal function G_i ;
  - T_i is the **transition function**
    - May contain exploitable structure across the sequence of tasks, especially object-centric structure;
  - γ is a **discount factor**
  - τ_i is task-specific **context information**
    - A vector of real numbers, possibly factored into object context: $\tau_i = \tau_1^i \times \ldots \times \tau_k^i$, for k objects.

AncoraSIR.com

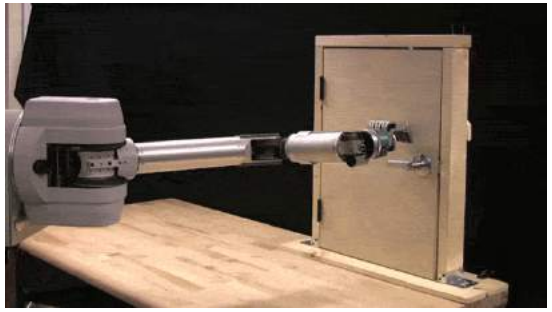
Learning to Open Doors

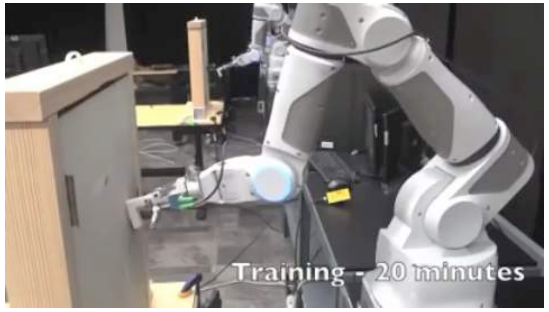
AncoraSIR.com

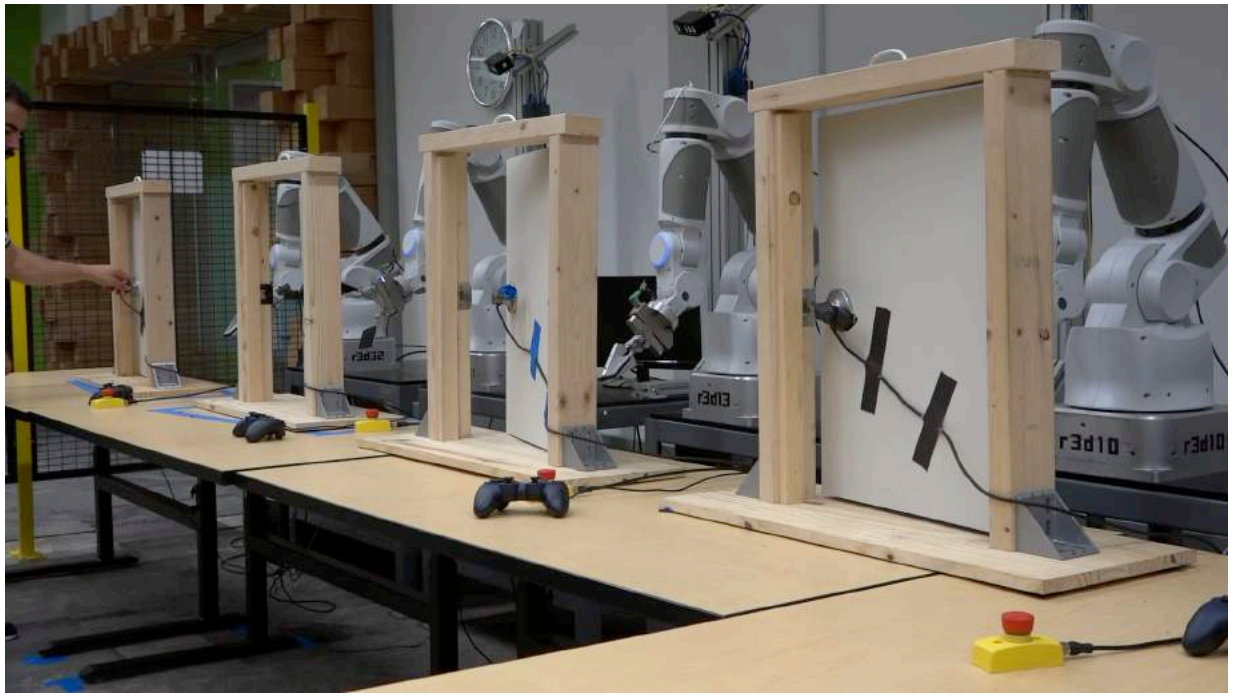
SUSTech
Southern University of Science and Technology

Example: Learning to Open Doors

Many doors …







A successful door opening policy

- Robust to many different
  - doors
  - lighting conditions
  - environment settings
- Quick, reliable, and safe

Subtask decomposition

- Turning the handle
- Pulling open the door

Sources: Google Brain, UC Berkley, Boston Dynamics

Approaches to Learn Openning Doors

Door Keypoint Detection, Model Based, and Reinforcement Learning based

- Door Keypoint Detection
  - locating the doorknob location, and axes of rotation,
  - then using motion planning to open the door

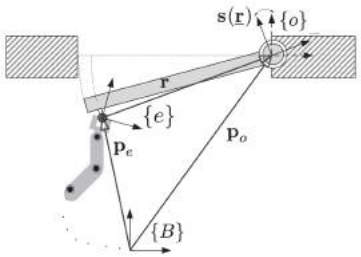
MANIPULATION TASK	DESCRIPTORS
OPEN A DOOR	1. LOCATION OF ROTATION AXIS OF HANDLE p_1 2. AXIS OF ROTATION v_1 3. LOCATION ALONG HANDLE AT WHICH TO PUSH DOWN p_2
PRESS AN ELEVATOR BUTTON	1. LOCATION OF BUTTON p_1 2. DIRECTION TO PRESS v_1
PULL OPEN A DRAWER	1. MIDPOINT OF THE HANDLE p_1 2. DIRECTION TO PULL v_1
TURN A THERMOSTAT KNOB	1. MIDPOINT OF THE KNOB p_1 2. AXIS OF ROTATION v_1

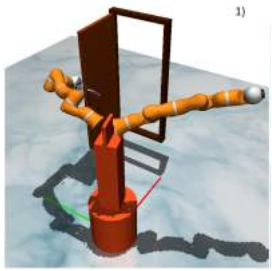
sci-hub.si/10.1109/IROS.2010.5649847

sci-hub.si/10.1109/iros.2012.6385835

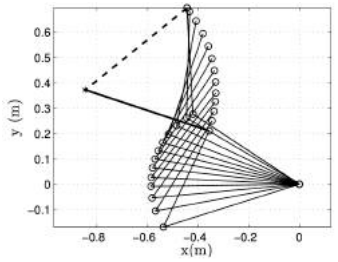
sci-hub.si/10.1109/icar.2017.8023522

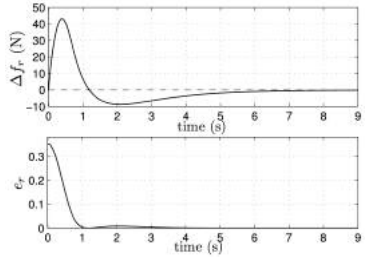
- Model Based
  - Creating a model of the door
  - Learning the kinematics of doors





- Reinforcement Learning based
  - A growing interest of research





Algorithm 1: door opening learning algorithm

Input: Initial position of the door handle
initialize control gains $\mathbf{K}_p$, $\mathbf{K}_o$
initialize $\boldsymbol{F}(k) = 0$, $\boldsymbol{M}(k) = 0, k = 1, \ldots T$
Output: Learned policy ($\boldsymbol{F}(k)$, $\boldsymbol{M}(k)$)

1 **for** $l = 1, \ldots,$ *max learning cycles* **do**
2 **for** $k = 1, \ldots, T$ **do**
3 **if** *robot is moving* **then**
4 calculate directions $\boldsymbol{d}_p$ and $\boldsymbol{d}_o$ (5,10)
5 calculate $\boldsymbol{F}(k)$ and $\boldsymbol{M}(k)$ (6,11)
  else
6 calculate auxilary phase variable s_a (22)
7 calculate $\boldsymbol{F}(s)$ and $\boldsymbol{M}(s)$ learned so far (15,16)
8 add noise $\boldsymbol{F}(k) = \boldsymbol{F}(s) + \mathcal{N}(0, \sigma_p^2)$
9 add noise $\boldsymbol{M}(k) = \boldsymbol{M}(s) + \mathcal{N}(0, \sigma_o^2)$
10 calculate and apply joint torques to the motors (12)
11 collect intermediate cost $c_i(k)$
12 collect terminal cost c_t
13 calculate $\boldsymbol{w}_{f,l}$ and $\boldsymbol{w}_{m,l}$ from $\boldsymbol{F}(k), \boldsymbol{M}(k)$ (19)
14 save data in the importance sampler
15 estimate new optimized weights $\boldsymbol{w}_{f,l+1}$ and $\boldsymbol{w}_{m,l+1}$ using PI^2 (14)
16 update search noise variance $\sigma_p^2 = \alpha\sigma_p^2$; $\sigma_o^2 = \alpha\sigma_o^2$;

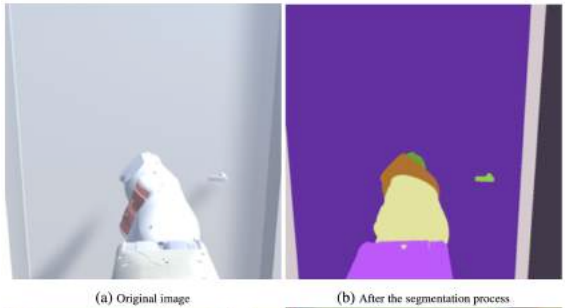
AncoraSIR.com

Domain Randomization in Reinforcement Learning

Bridge the reality gap between simulation and the real world

- **Drawbacks in Reinforemenct Learning**
  - sample inefficiency, the curse of dimensionality, and
  - the **reality gap** between simulators and the real world
- **Assumption**
  - Hard to model the real world perfectly, but easy to create many different simulations that approximate the real world.
- **Proposal**
  - It is possible to ensemble a variety of simulator environments with different visual or physical properties to generalize to a domain that overlaps the real world

Examples of the post processing that can be apply by Unity



Top row shows the function to make semantic segmentation labels.

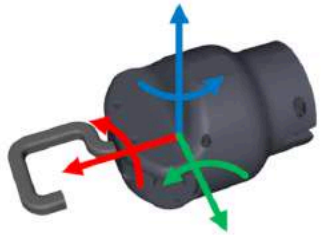
Bottom row shows the image when the Gaussian noise, camera distortion, and chromatic aberration are added.

OpenAI Gym

(a) 8DoF BLUE with grippers

(b) 7DoF BLUE with hook

(c) 7DoF Floating gripper

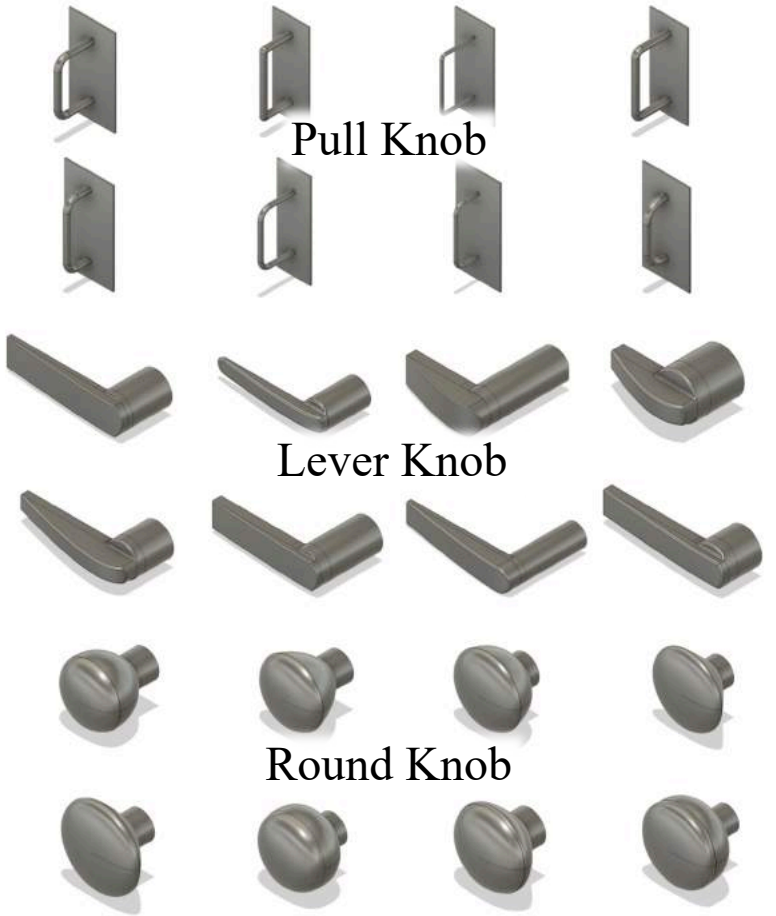


(d) 6DoF Floating hook

List of Randomized Parameters in the Door World

Door

Randomization Parameters		Scalling Factor Range	Unit
Door Physical parameters			
Wall Property	wall location	y-axis: Uniform[-200, 200]	mm
Door Frame Joint Property	door frame damper	Uniform[0.1, 0.2]	-
	door frame spring	Uniform[0.1,0.2]	-
	door frame frictionloss	Uniform[0, 1]	-
Door Property	door height	Uniform[2000, 2500]	mm
	door width	Uniform[800, 1200]	mm
	door thickness	Uniform[20, 30]	mm
	knob height	Uniform[950, 1050]	mm
	knob horizontal location ratio	Uniform[0.10, 0.20]	-
	door mass	Uniform[22.4, 76.5]	kg (Based on MDF density)
	hinge position	left hinge/right hinge	-
	opening direction	push/pull	-
Knob Door Joint Property	knob door damper	Uniform[0.1, 0.2]	-
	knob door spring	Uniform[0.1, 0.15]	-
	knob door frictionloss	Uniform[0, 1]	-
	knob rot range	Uniform[75, 80]	degree
Knob Property	knob mass	Uniform[4, 7]	fg
	knob surface friction	Uniform[0.50, 1.00]	-



STL models

AncoraSIR.com

List of Randomized Parameters in the Door World

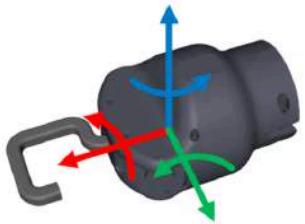
Robot

Randomization Parameters		Scalling Factor Range	Unit
Robot Physical Parameters			
Robot Property	arm joint damping	Uniform[0.1, 0.3]	-

(a) 8DoF BLUE with grippers

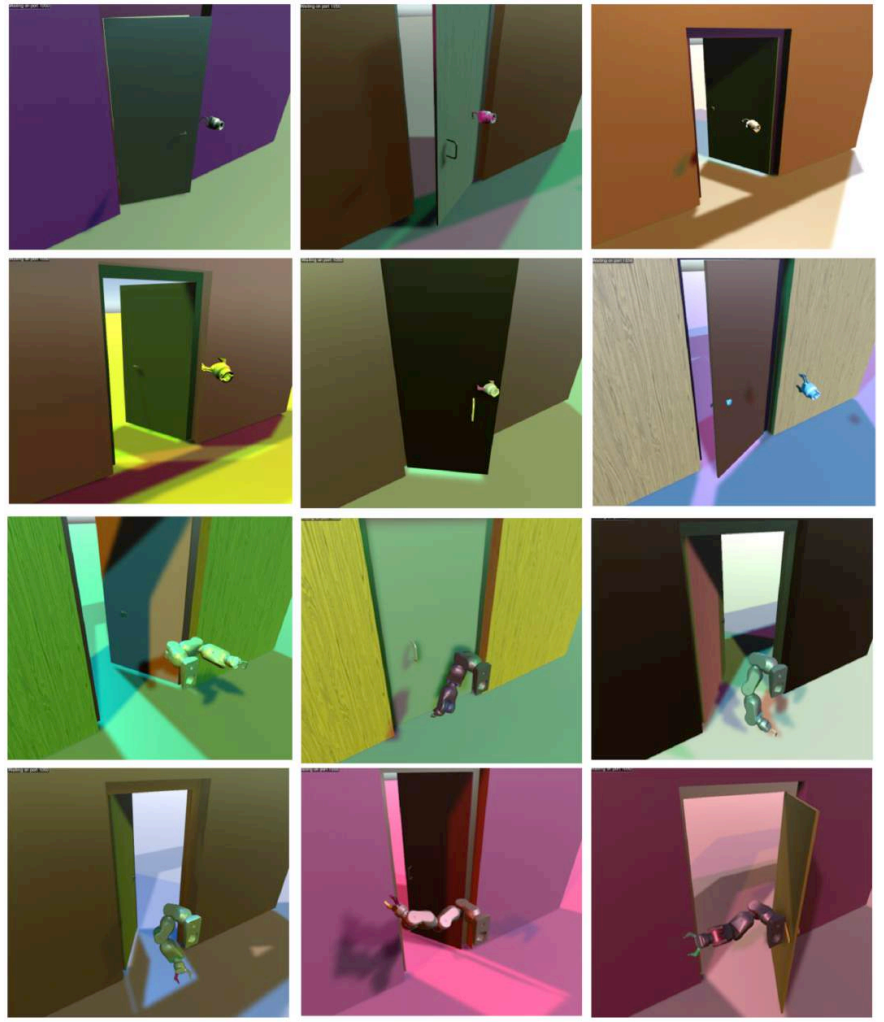
(b) 7DoF BLUE with hook

(c) 7DoF Floating gripper



(d) 6DoF Floating hook

https://www.berkeleyopenarms.org/

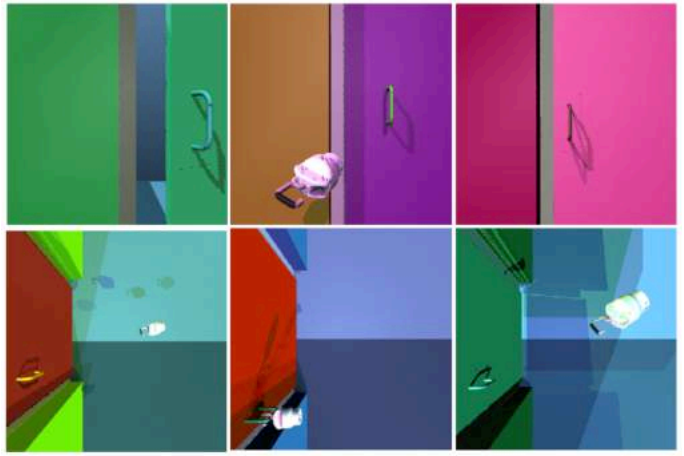


List of Randomized Parameters in the Door World

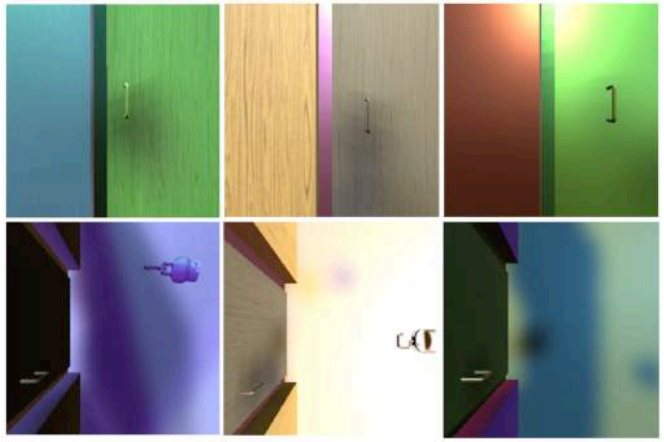
Vision

Randomization Parameters		Scalling Factor Range	Unit
Vision Parameters			
Lighting	light number	Uniform[2-6]	-
	light diffuse	Uniform[0.0, 1.0]	RGBA
	light position	x:Uniform[0.0, 5] y:Uniform[-5, 5] z:Uniform[3, 7]	m
	light direction	x:Uniform[-0.5, 0.5] y:Uniform[-0.5, 5.0] z:Uniform[-0.5, -0.25]	rad
Wall Material	wall shininess	Uniform[0.01, 0.50]	-
	wall specular	Uniform[0.01, 0.50]	-
	wall rgb1	Uniform[0.0, 1.0]	RGBA
Frame Material	frame shininess	Uniform[0.01, 0.70]	-
	frame specular	Uniform[0.01, 0.80]	-
	frame rgb1	Uniform[0.0, 1.0]	RGBA
Door Material	door shininess	Uniform[0.01, 0.30]	-
	door specular	Uniform[0.01, 0.80]	-
	door rgb1	Uniform[0.0, 1.0]	RGBA
	door rgb2	Uniform[0.0, 1.0]	RGBA
Doorknob Material	knob shininess	Uniform[0.50, 1.00]	-
	knob specular	Uniform[0.80, 1.00]	-
	knob rgb1	Uniform[0.0, 1.0]	RGBA
Robot Material	robot shininess	Uniform[0.01, 0.70]	-
	robot specular	Uniform[0.01, 0.80]	-
	robot rgb	Uniform[0.0, 1.0]	RGBA

MuJoCo API



(a) MuJoCo rendered images



(b) Unity rendered images

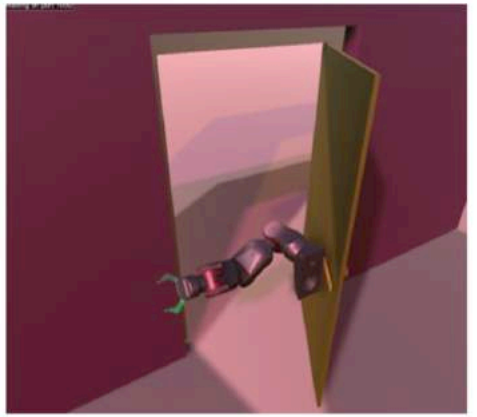
Unity can render realistic shadows and detailed textures.

SUSTech
Southern University of Science and Technology

The ***Shaped*** Reward Function for DoorGym

Use simple and readily available observations from the environment

- Elements of an observation
  - The position and the velocity of each robot joint
  - The position of the door-knob
    - *Can be obtained directly from the simulator, or using a 256x256 RGB image and vision network*
  - The position of the end-effector in world coordinates
- Policy actions
  - Force for linear actuators and torque for rotational actuators, which can be configured to use position control
  - The size of the action space corresponds to the # DoF of each robot



Subscript indicate the value at time t

To minimize (-)

To maximize (+)

The Reward Function to incentivize the agent to open the door

$$r_t = -a_0 d_t - a_1 log(d_t + \alpha) - a_2 o_t - a_3 \sqrt{u_t^2} + a_4 \phi_t + a_5 \psi_t$$

The distance between the fingertip of the end-effector and the center coordinate of the doorknob

To give a precision when the agent get close to a target ($\alpha = 0.005$)

The difference between the current fingertip orientation of the robot and the ideal orientation to hook/grip the doorknob

The control input to the system

The angle of the door

The angle of the door-knob

Quiz:
The robot tries to open the door ***before or after*** it hooks or rotates the door knob ?

$$a_{0,1,2,3} = 1$$
$$a_4 = 30$$
$$a_5 = 50$$

AncoraSIR.com

The *Unshaped* Reward Function for DoorGym

Other forms of evaluation

- We define a successful attempt as the robot opening the door at least 0.2 radians within 10 seconds.
  - For attempt i, $\mathbb{1}_i = \begin{cases} 1 & \text{if } \phi > 0.2 \text{ rad and t} < 10 \\ 0 & \text{otherwise} \end{cases}$
  - Two unshaped reward function (*measured over 100 attempts*)

The average success rate of opening a door

$$r_{\text{AT}} = \frac{1}{n}\sum_{i=1}^{n} t_i$$

The average time to open the door

$$r_{\text{ASR}} = \frac{1}{100}\sum_{i=1}^{100} \mathbb{1}_i$$

t_i: the time to completion for successful attempts
n: the number of successful attempts

To minimize (-)

To maximize (+)

The Reward Function to incentivize the agent to open the door

$$r_t = -a_0 d_t - a_1 log(d_t + \alpha) - a_2 o_t - a_3 \sqrt{u_t^2} + a_4 \phi_t + a_5 \psi_t$$

Subscript indicate the value at time t

d_t: The distance between the fingertip of the end-effector and the center coordinate of the doorknob

α: To give a precision when the agent get close to a target ($\alpha = 0.005$)

o_t: The angle of the door

u_t: The control input to the system

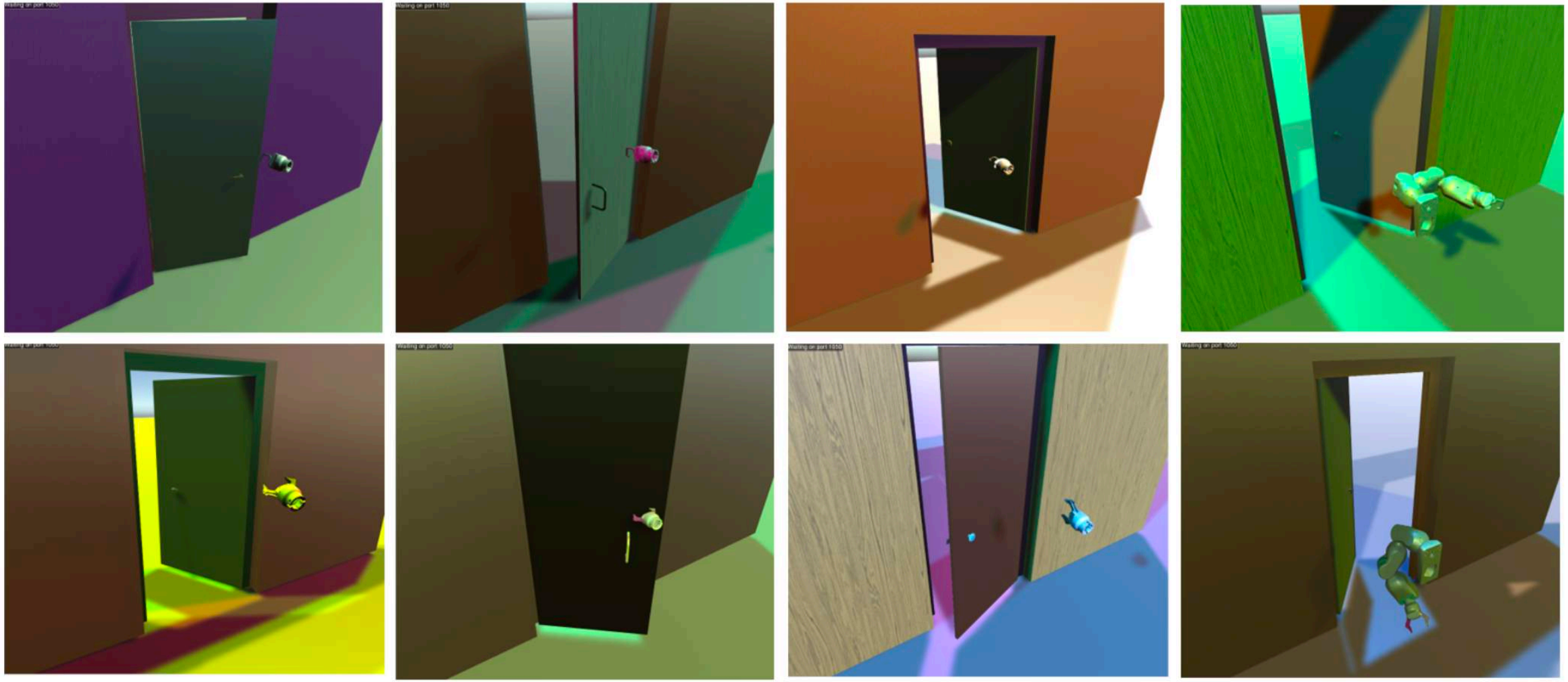
ψ_t: The angle of the door-knob

$a_{0,1,2,3} = 1$
$a_4 = 30$
$a_5 = 50$

AncoraSIR.com

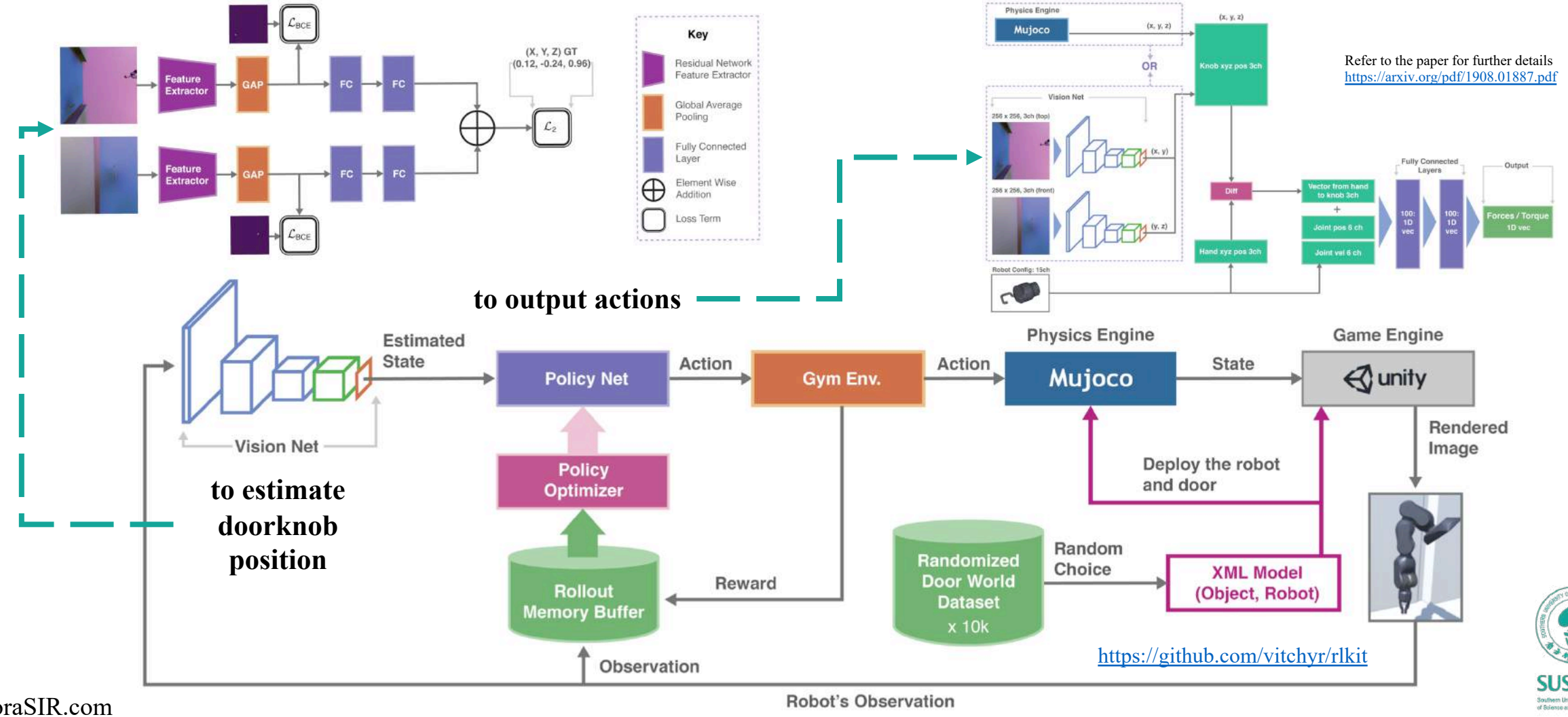
Environment Summary

36 possible environment combinations with 3 grippers, 6 robots, and 2 directions (push/pull)



The Door Opening Agent

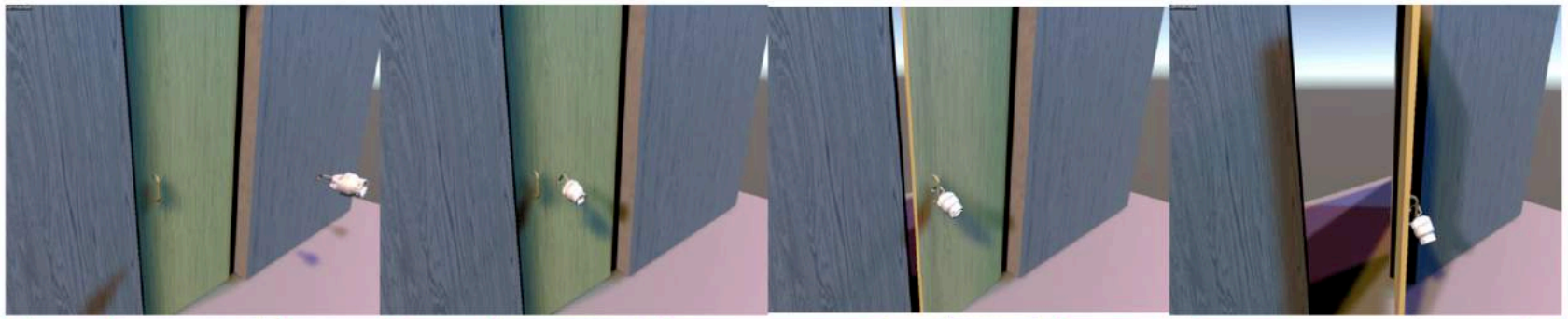
*An on-policy (**P**roximal **P**olicy **O**ptimization) and off-policy (**S**oft **A**ctor **C**ritic) update algorithm*



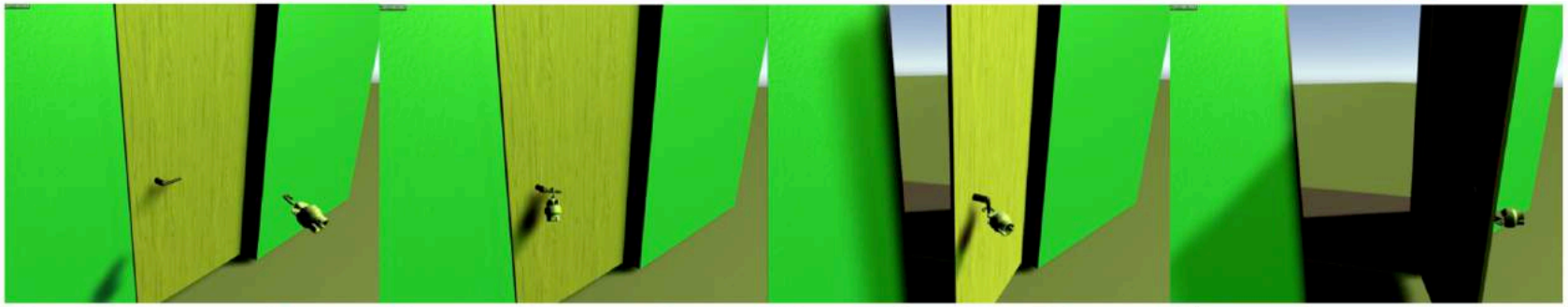
AncoraSIR.com

Experiments

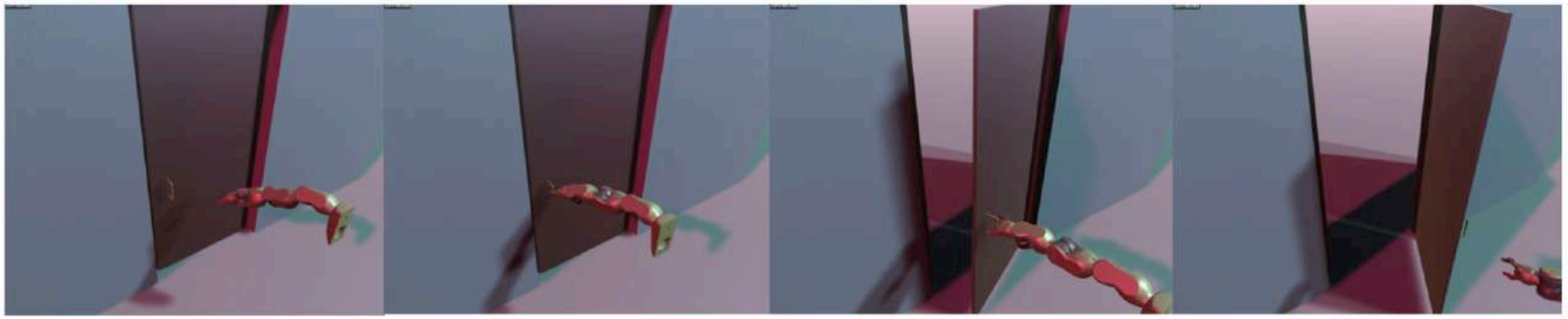
The agent has 10.2sec to try open the door for each of the 100 test door-worlds



Task 1 has a **pull knob** environment with ***floating hook***



Task 2 has a **lever knob** with ***floating hook***



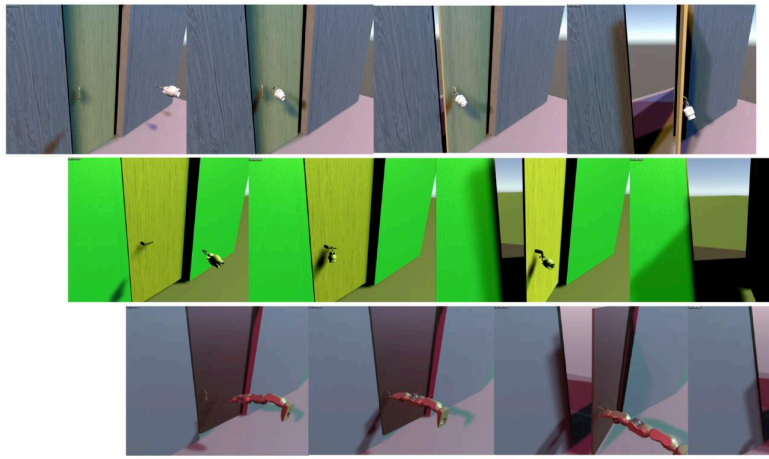
Task 3 has a **pull knob** with the ***BLUE-with-gripper platform***

AncoraSIR.com

Results

Average Success Ratio and Average Time to Open measured in seconds

		Task 1		Task 2		Task 3	
		r_{ASR}	r_{AT}	r_{ASR}	r_{AT}	r_{ASR}	r_{AT}
PPO	Ground Truth Position	0.73	3.73	0.68	4.12	0.71	5.13
	Ground Truth Position + N(0, σ)	0.70	4.28	0.54	4.11	0.65	4.73
	Vision Network Estimated Position	0.23	5.00	0.29	3.83	0.57	5.12
SAC	Ground Truth Position	0.23	4.29	0	N/A	0.33	5.02
	Ground Truth Position + N(0, σ)	-	-	-	-	-	-
	Vision Network Estimated Position	0	N/A	0	N/A	0.05	5.06



Task 1 has a **pull knob** environment with ***floating hook***

Task 2 has a **lever knob** with ***floating hook***

Task 3 has a **pull knob** with the ***BLUE-with-gripper platform***

AncoraSIR.com

- PPO (~70%) has a much higher successrate than SAC
- SAC shows better exploration and faster convergence, but PPO shows better exploitation in trade-off of its training speed
- Success rates of both algorithm decrease
  - When adding gaussian noise to the door knob position
  - When the door knob position information comes from the vision network
- Position estimation in the 3D space of the door knob is **extremely important** for the door opening task.

SUSTech
Southern University of Science and Technology

Thank you~

songcy@sustech.edu.cn

AncoraSIR.com