UC San Diego

Interfacing Perception and Policy Learning Algorithms for Embodied AI

Jiayuan Gu

About Me

- Fifth-year Ph.D. student advised by Prof. Hao Su at the University of California San Diego (UCSD)
- My homepage: http://cseweb.ucsd.edu/~jigu/

Research Topics (Selected Publications)

2D vision

- Relation Networks for Object Detection (CVPR 2018 oral)

3D vision

- Weakly-supervised 3d shape completion in the wild (ECCV 2020 spotlight)
- Multi-view PointNet for 3D Scene Understanding (ICCV-W 2019)

Policy Learning

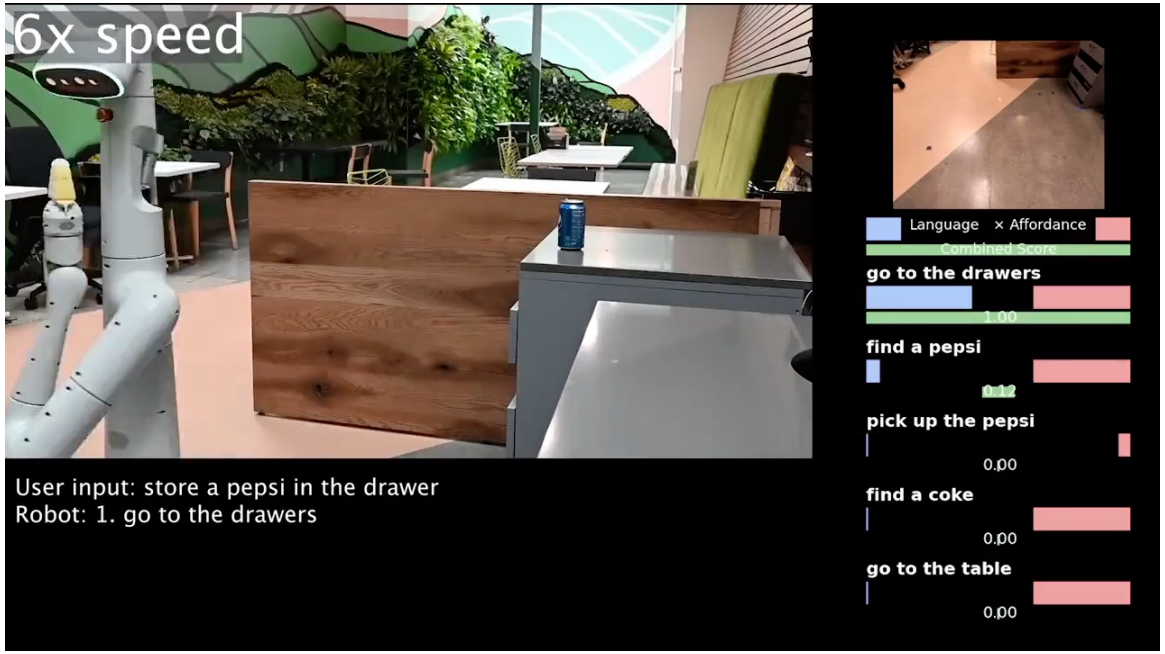
- Refactoring Policy for Compositional Generalizability using Self-Supervised Object Proposals (NeurIPS 2020)
- Multi-skill Mobile Manipulation for Object Rearrangement (ICLR 2023 spotlight)
- ManiSkill2: a Unified Benchmark for Generalizable Manipulation Skills (ICLR 2023)

Policy Learning

Topics: visuomotor mobile manipulation and building realistic simulation environments

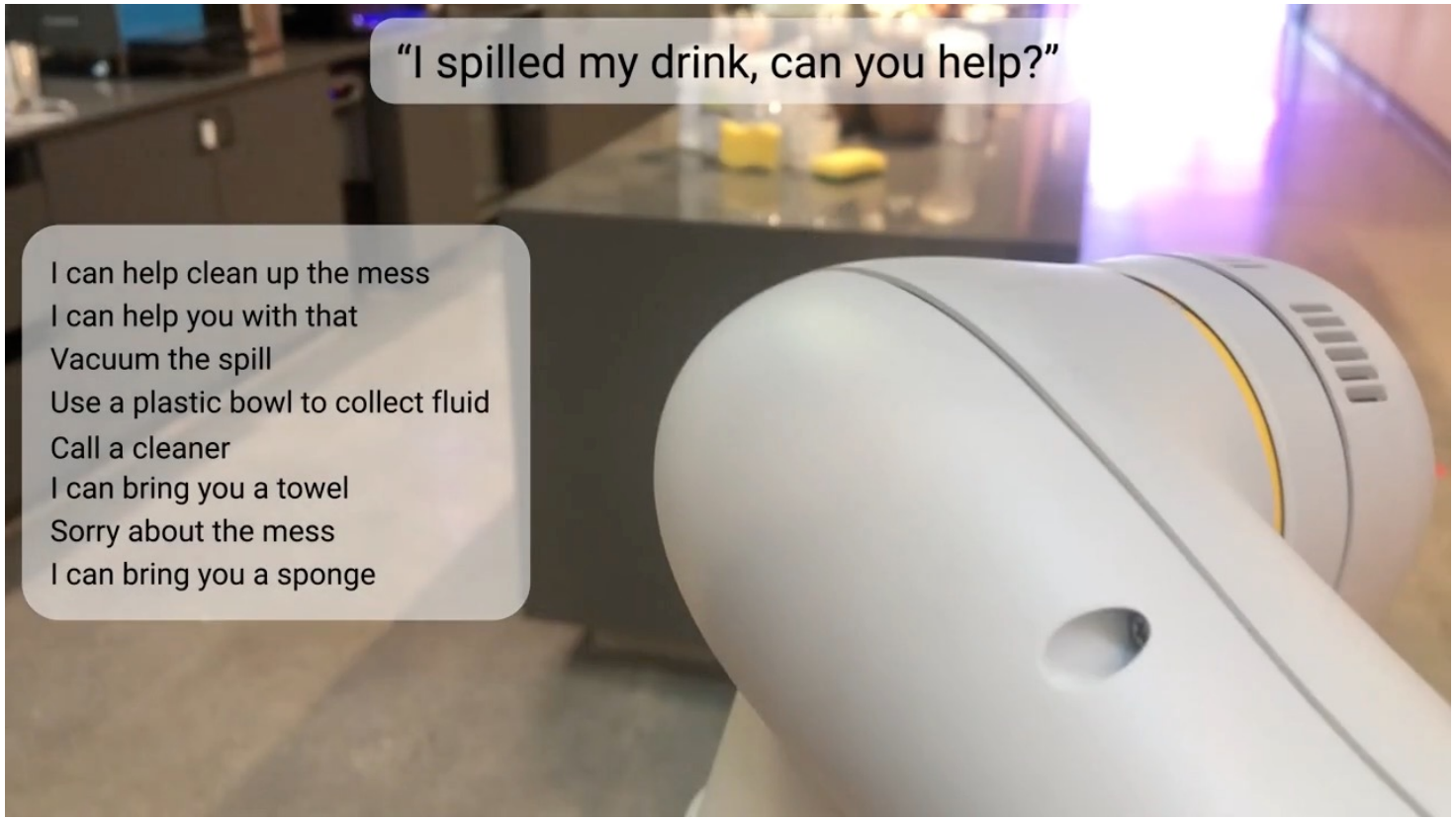
Tackle Daily Chores by Skills

M3 (our work, ICLR 23)



SayCan (Google, CoRL 22)

However, generalizable skills are not ready



SayCan (Google, CoRL 22)

Limitation: Skills are limited to certain objects and layouts

Deep Learning Paradigm: Big Data and Models

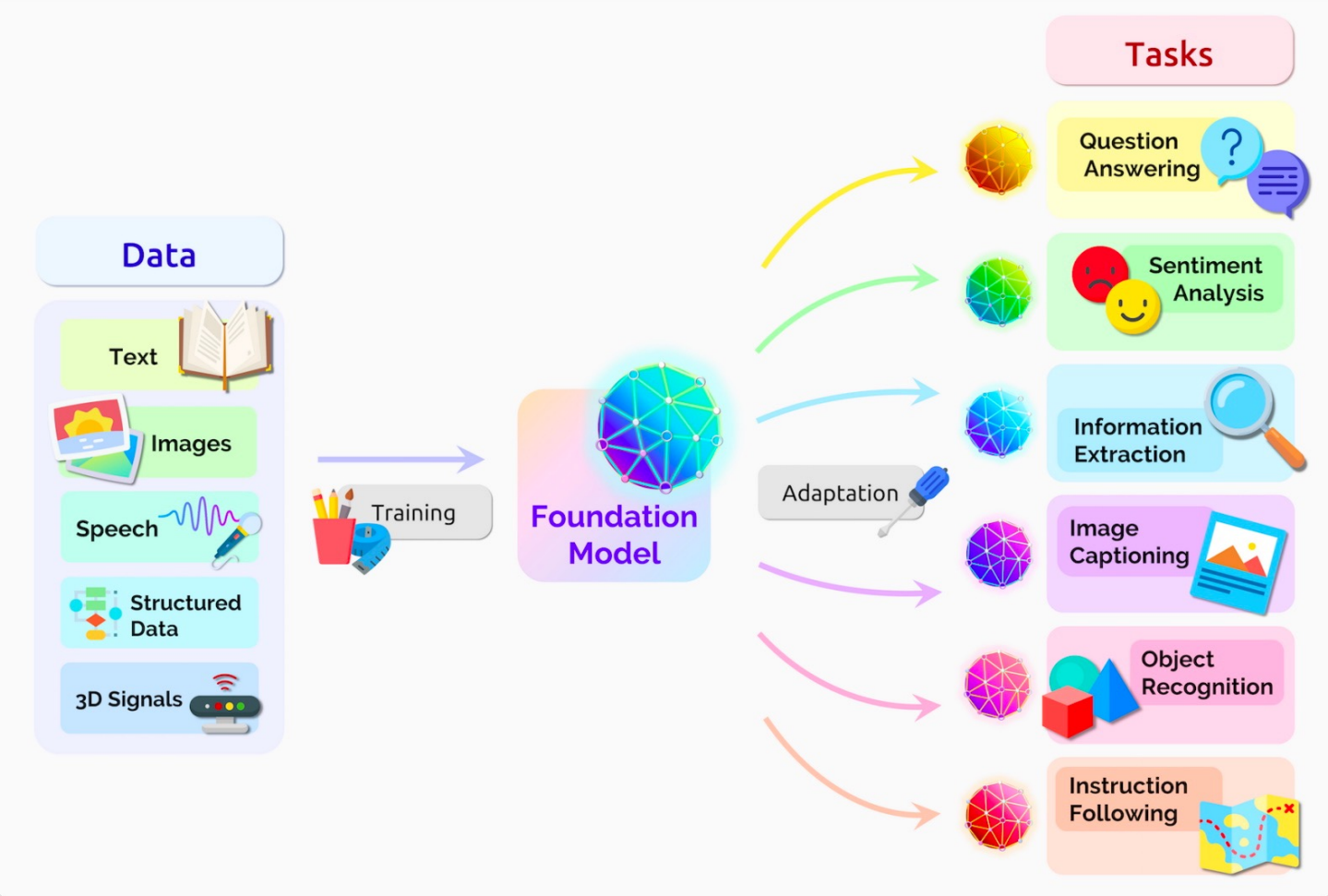
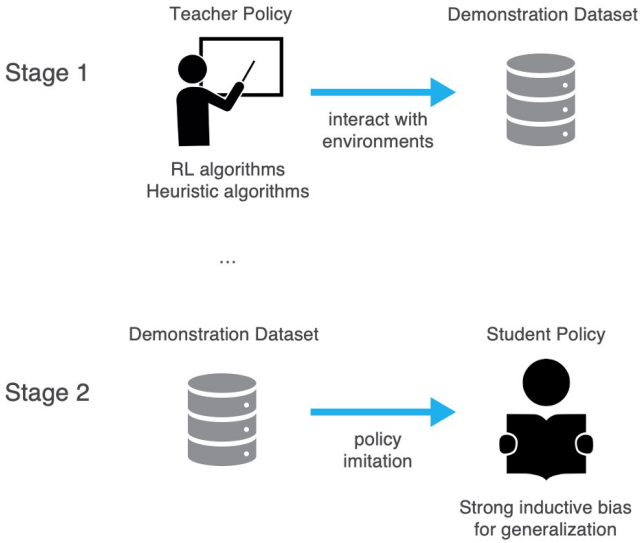


Image source: https://blogs.nvidia.com/blog/2022/03/25/what-is-a-transformer-model/

Data and Model for Embodied AI



Acquire generalizable skills from offline data (policy refactorization)

Build environments to acquire data (ManiSKill2)

Design and learn generalizable visuomotor skills (M3)

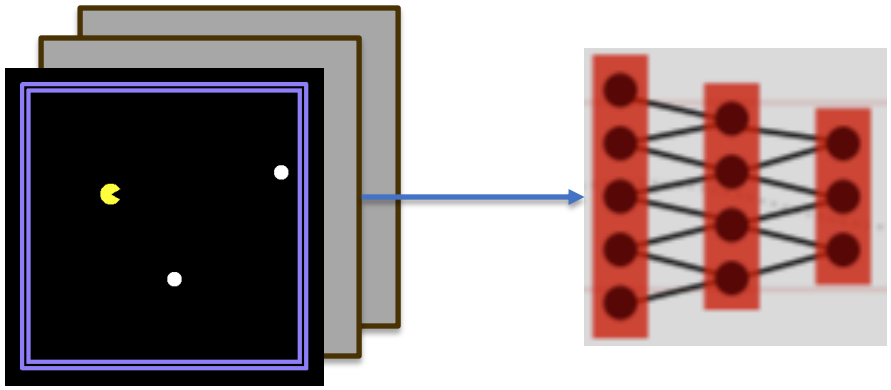
Learning Policy with Compositional Generalizability using Self-Supervised Object Proposals

Tongzhou Mu*, Jiayuan Gu*, Zhiwei Jia, Hao Tang, Hao Su

UC San Diego, Shanghai Jiaotong University

NeurIPS 2020

Challenges of Classical RL

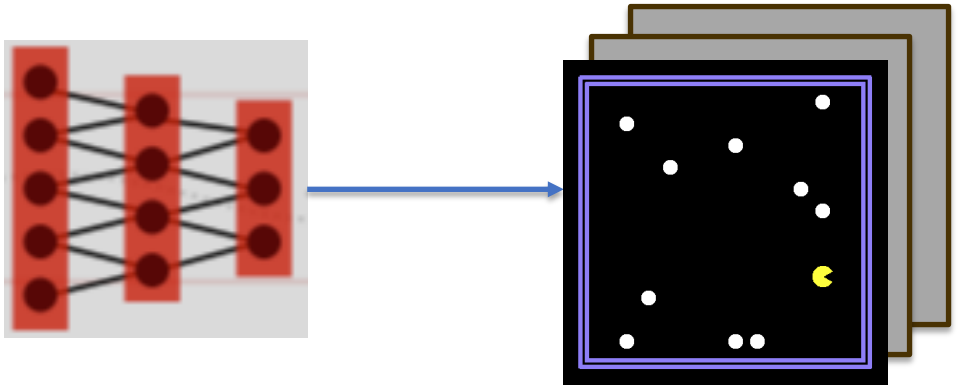


Training environments

Neural Network

Optimization Challenge

Challenges of Classical RL

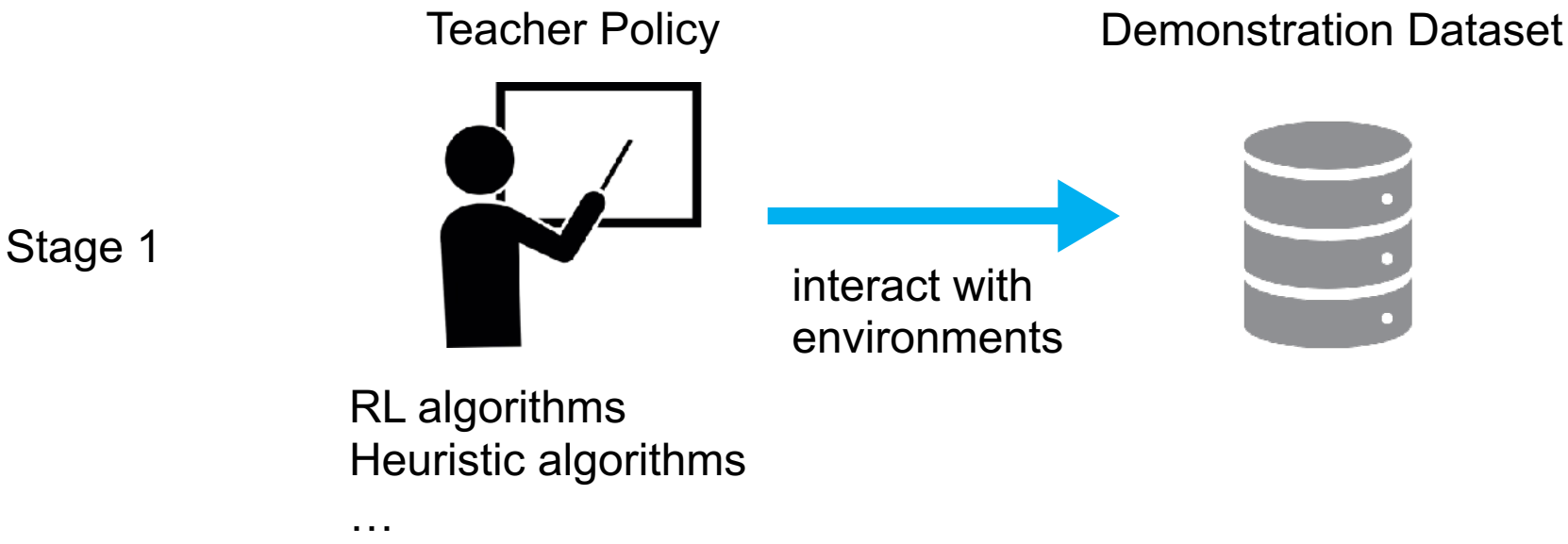


Neural Network

Test environments

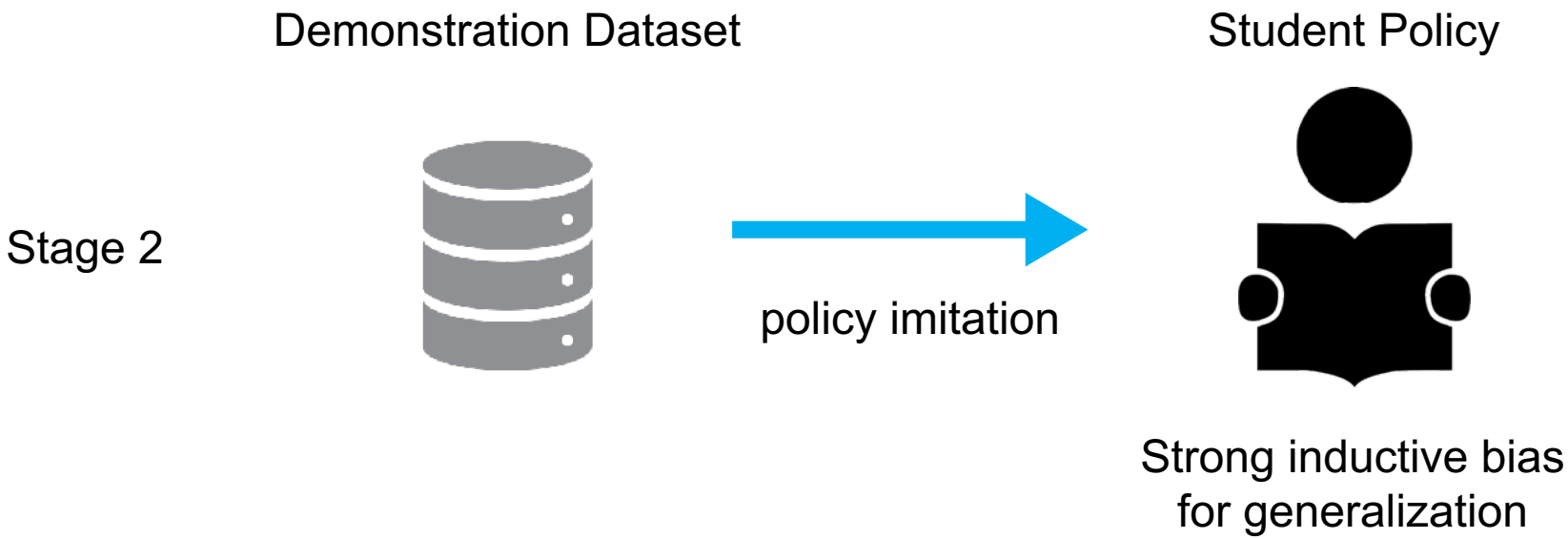
Generalization Challenge

Refactorization



Demonstration Acquisition without Generalizability Concerns

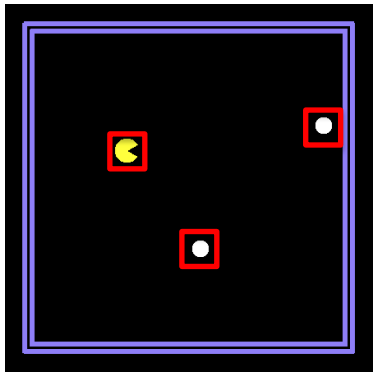
Refactorization



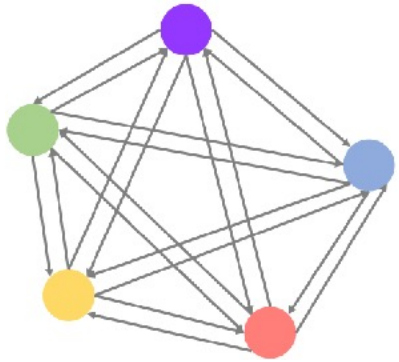
Refactorize Demonstration into Generalizable Policy

Object-centric Policy

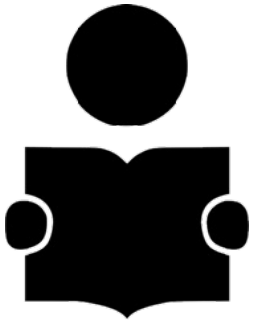
Self-Supervised
Object Detector



Graph Neural Network



Student Policy



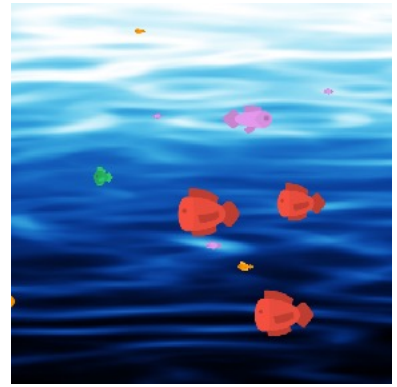
Strong inductive bias:
Object-centric Scene Graph

Experiments

- Flexible number of objects
- Random object arrangement
- Composition of foreground/background

Multi-MNIST

FallingDigit



BigFish

Multi-MNIST

Object-centric graph can be a strong inductive bias for compositional generalizability




Training Set

Test Set

Method	Train Acc	Test Acc
CNN	90.5(2.9)	12.0(2.1)
Relation Net	96.4(0.8)	8.4(4.7)
Ours	80.2(0.2)	**51.2(1.2)**

FallingDigit

The student network with object-centric graph inductive bias can refactorize the teacher policy into a compositional generalizable policy

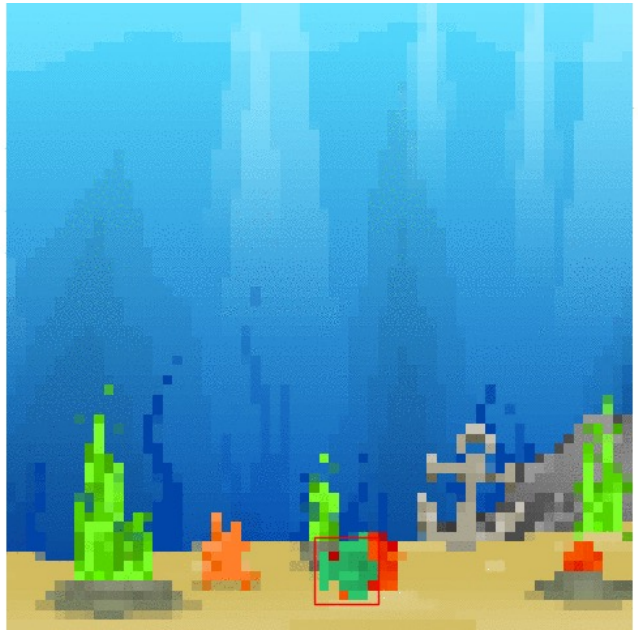
Train on 3 digits

CNN-based RL policy fails to generalize to 9 digits

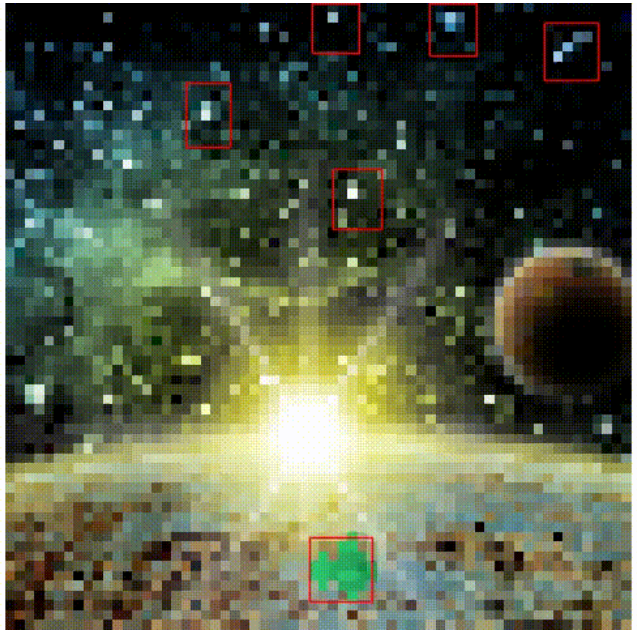
GNN-based refactorized policy generalizes to 9 digits

BigFish

More robust to different composition of foreground and background



Training Environments



Test Environments

Takeaway

- Refactorization through a proper student network with strong inductive bias can ease optimization and achieve compositional generalizability

- Divide-and-conquer is a strategy towards generalizable policies, which can also be used to collect data

ManiSkill2: A Unified Benchmark for Generalizable Manipulation Skills

Jiayuan Gu, Fanbo Xiang, et.al.

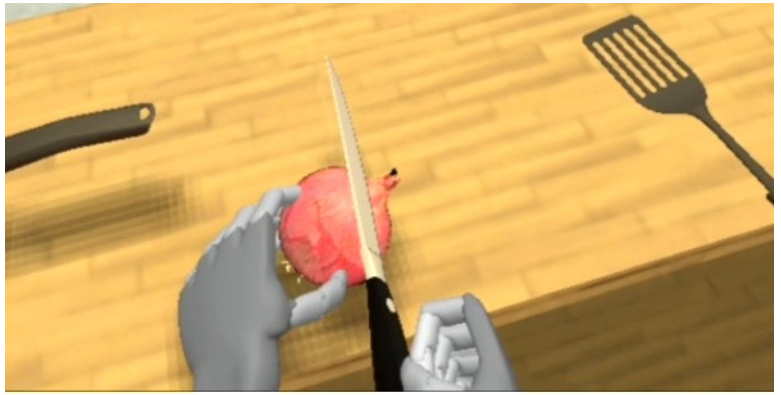
UC San Diego, Tsinghua University

ICLR 2023

Simplification in Prior Works

AI2THOR

Habitat 2.0



BEHAVIOR

Challenges: Generalization

- It is difficult to learn generalizable skills on **a large variety of objects** (topology and geometry variations)

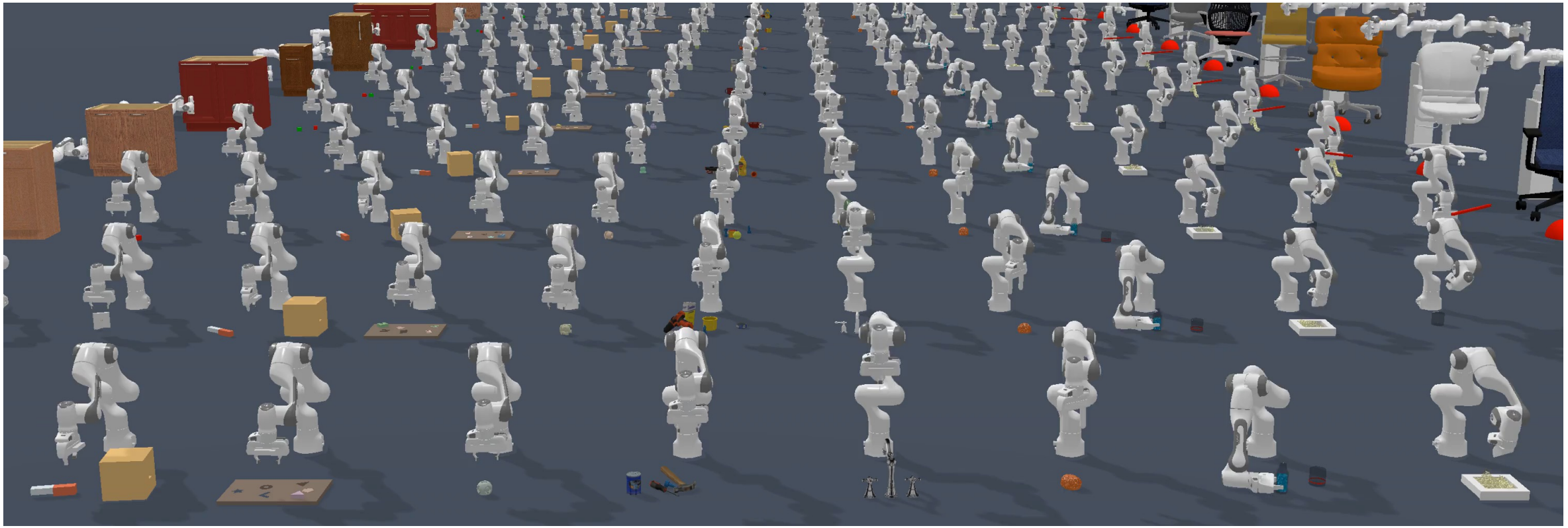



Generalizable Visuomotor Skills

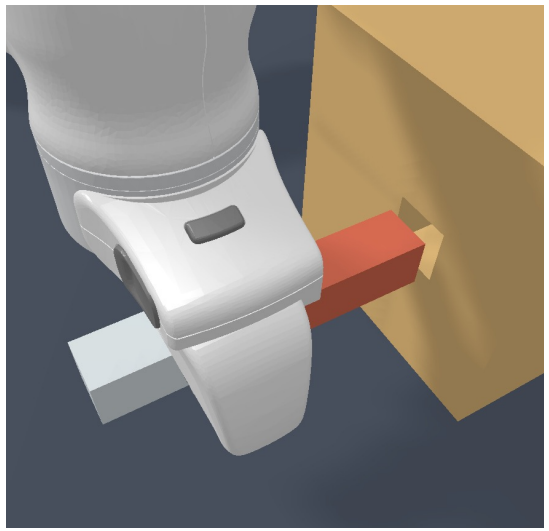
ManiSkill2

- Address challenges for generalizable manipulation skills
  - Large-scale demonstrations (4M+ frames)
  - Diverse assets (2000+)
- Features
  - Multiple types of manipulation tasks focusing on generalization
  - Conversion of demonstration action spaces
  - Highly efficient rigid-body environments for visual policy learning
  - A unified benchmark for a wide range of applications (RL, IL, TAMP, sense-plan-act)

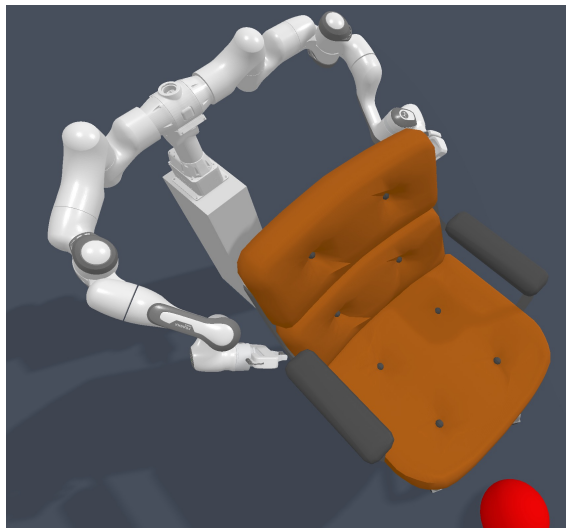
Large-scale Demonstrations and Diverse Assets



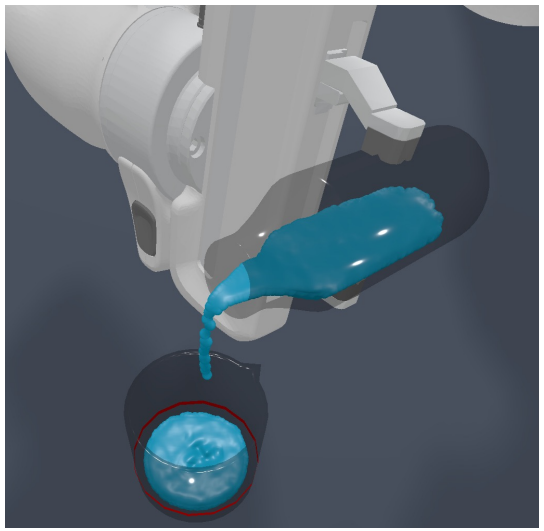
Feature 1: Multiple types of manipulation tasks focusing on generalization



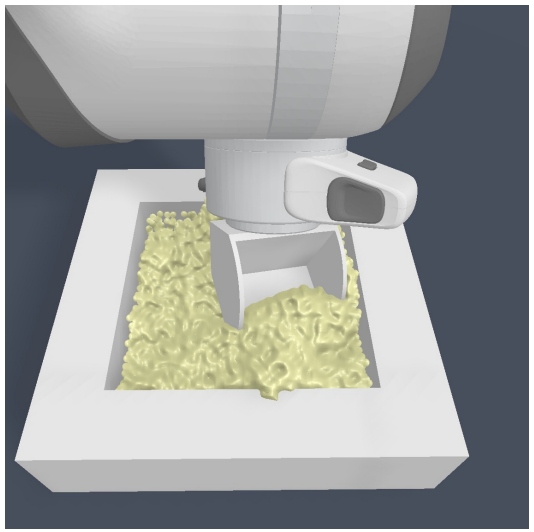
Single-arm
Stationary manipulation



Dual-arm
Mobile manipulation

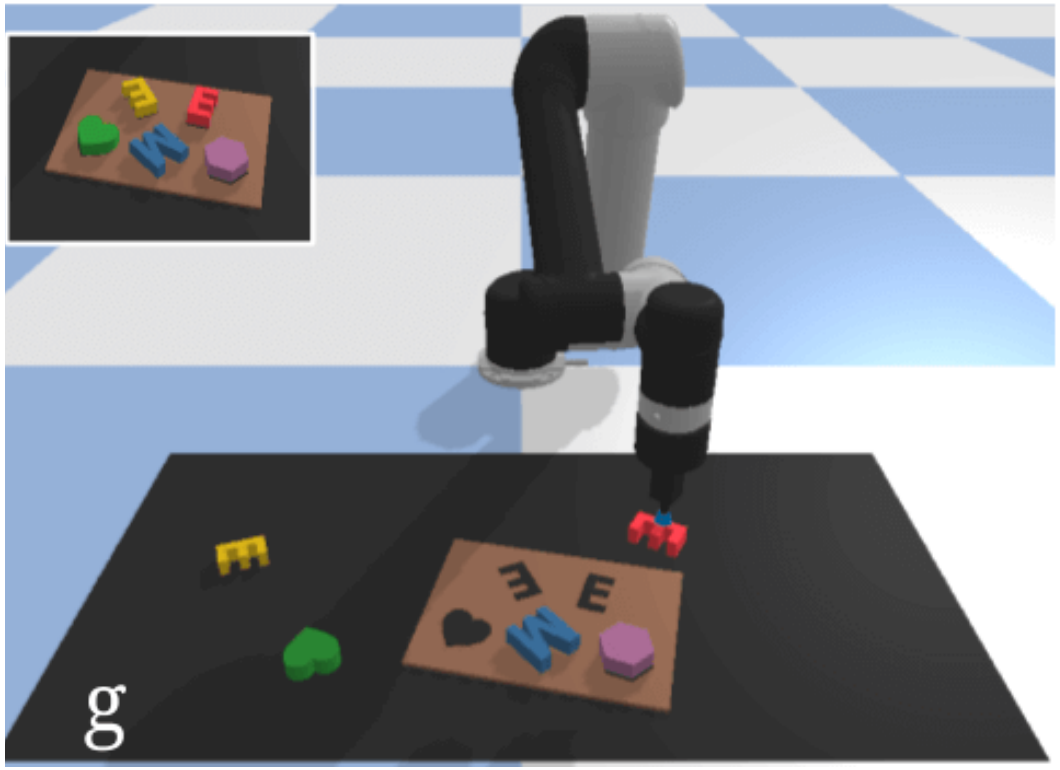


Soft-body



Soft-body

Example: Assembling Kits



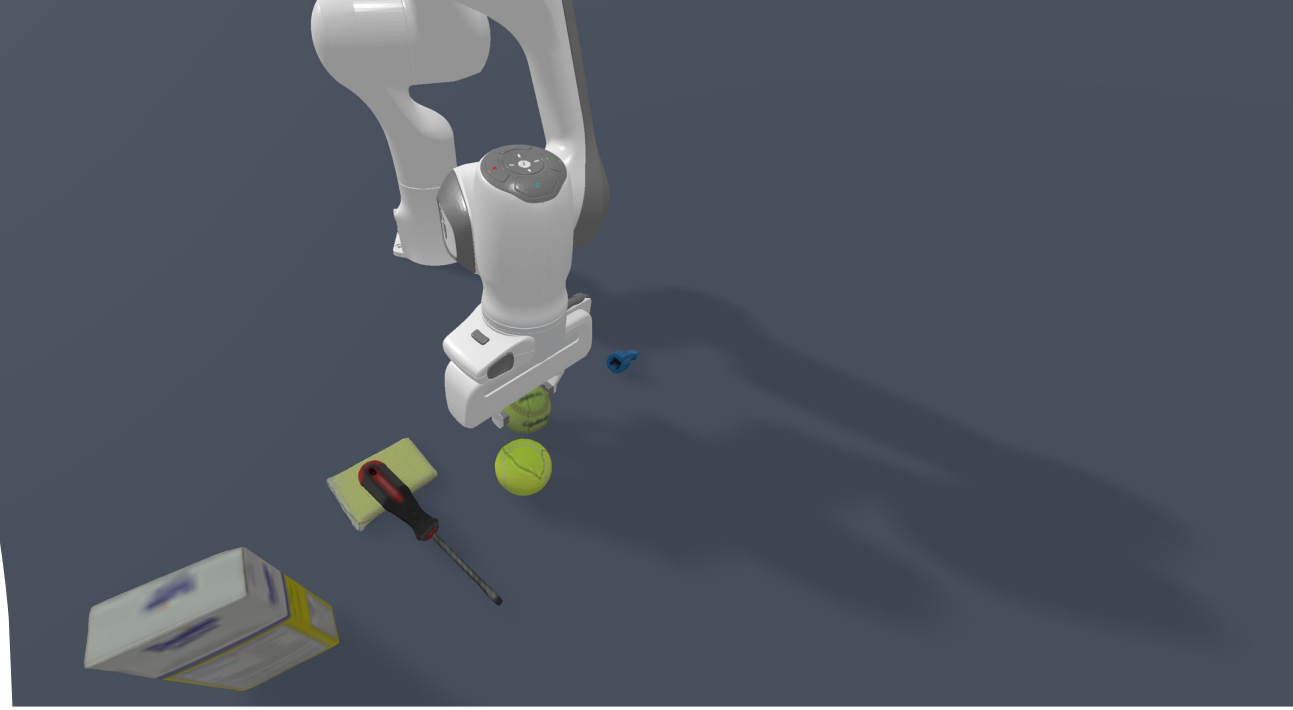
Prior work: The holes are not modeled

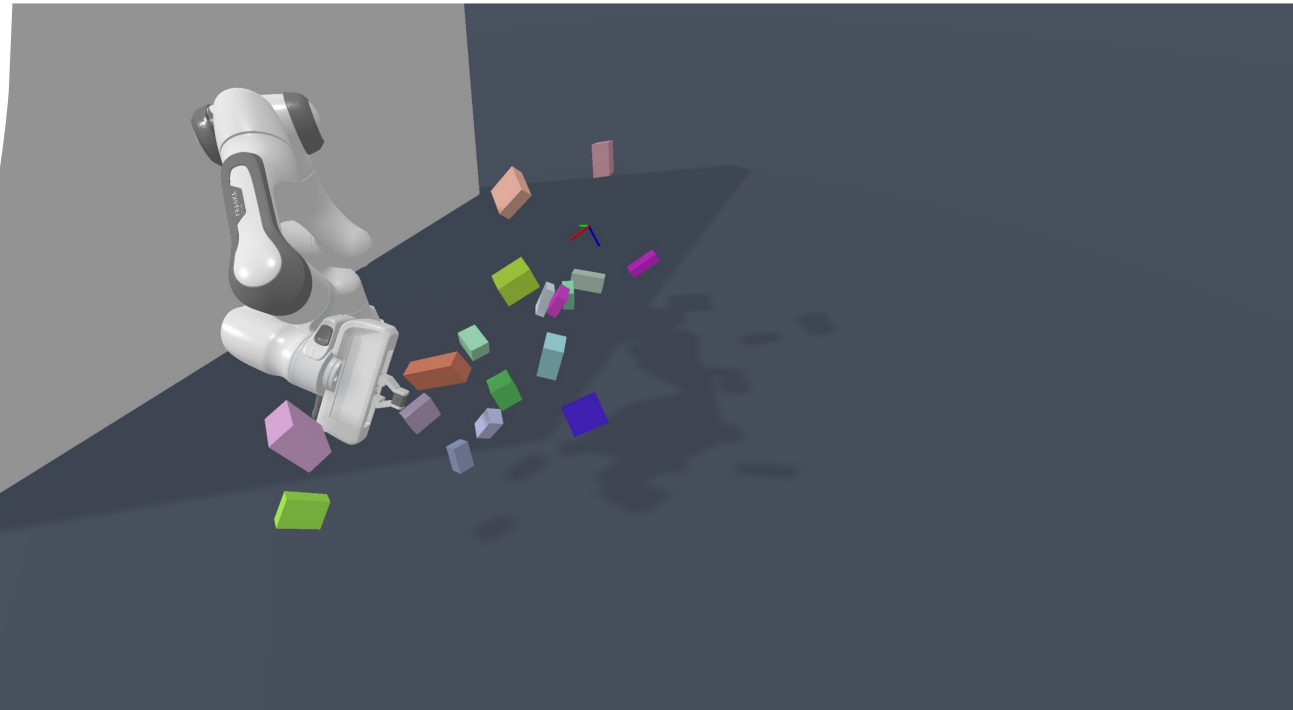
Ours: The board is carved to fit objects

Feature 2: Conversion of demonstration action spaces

- Different tasks are best solved with different controllers
  - Collision avoidance in motion planning requires **joint-space** controllers
  - Picking objects is simpler with **task-space** controllers

- We support converting the action space of demonstrations to a desired one



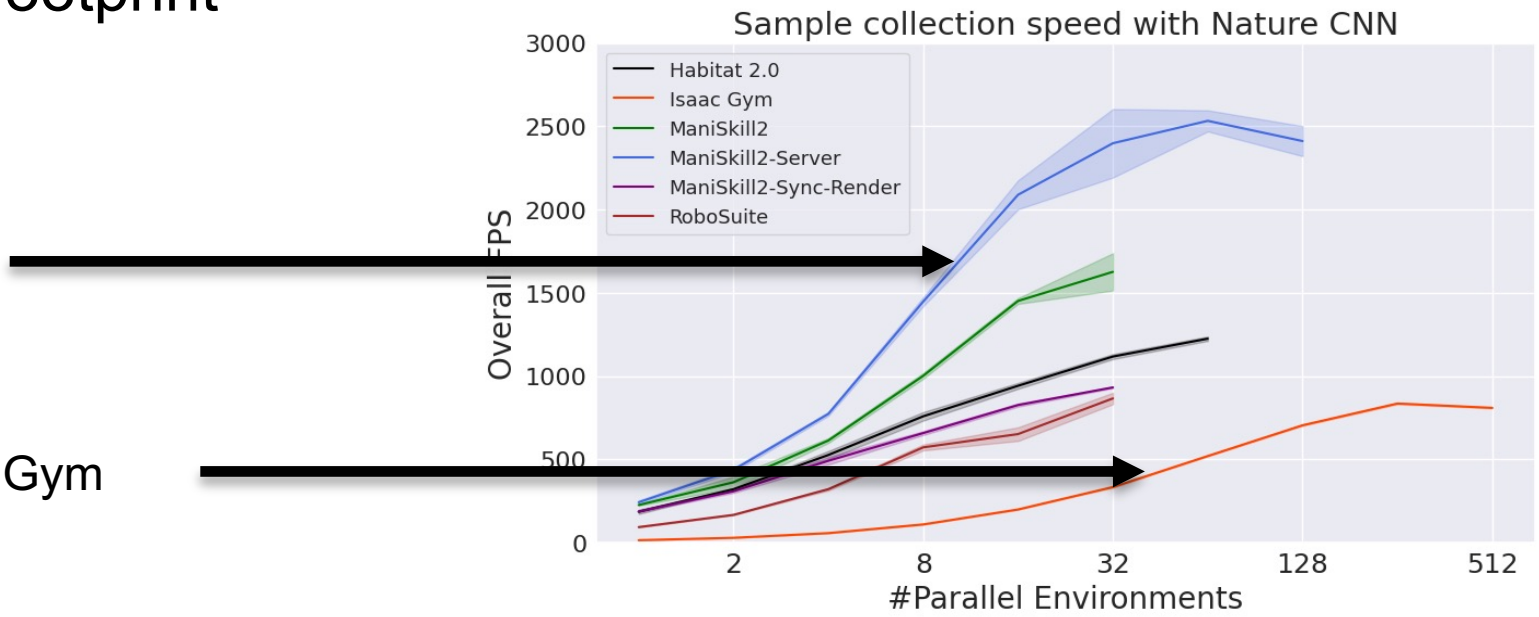


Feature 3: Highly efficient rigid-body environments for visual policy learning

- Present a highly efficient system with **asynchronous rendering** and **RPC-based (remote procedure call) render server**
  - Faster sample collection speed (2000+ FPS)
  - Lower memory footprint

ManiSkill2 render server (coming soon)

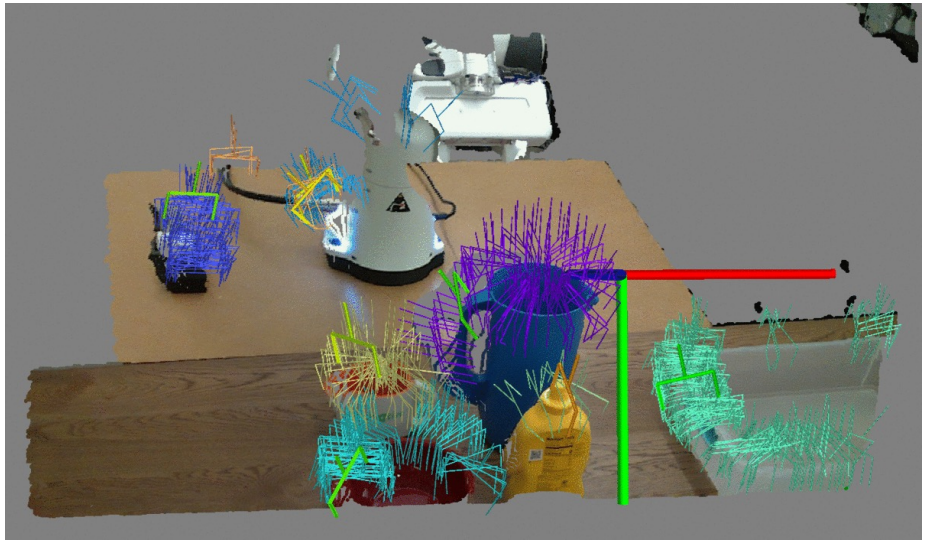
Issac Gym



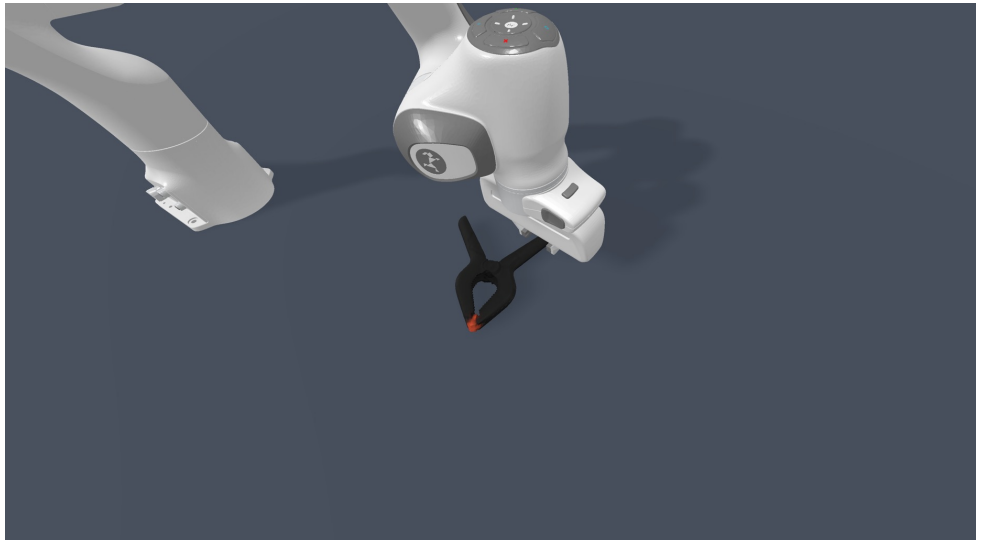
Feature 4: A unified benchmark for a wide range of applications

- Sense-plan-act (SPA)
- Reinforcement learning (RL)
- Imitation learning (IL)
- Sim2Real

Example 1: Contact-GraspNet



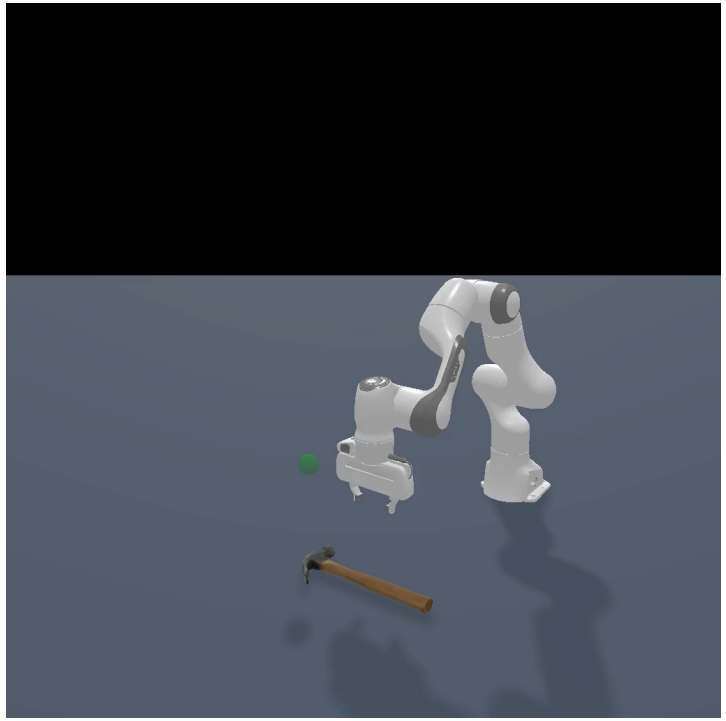
Contact-GraspNet



PickSingleYCB

Example 1: Contact-GraspNet

Success

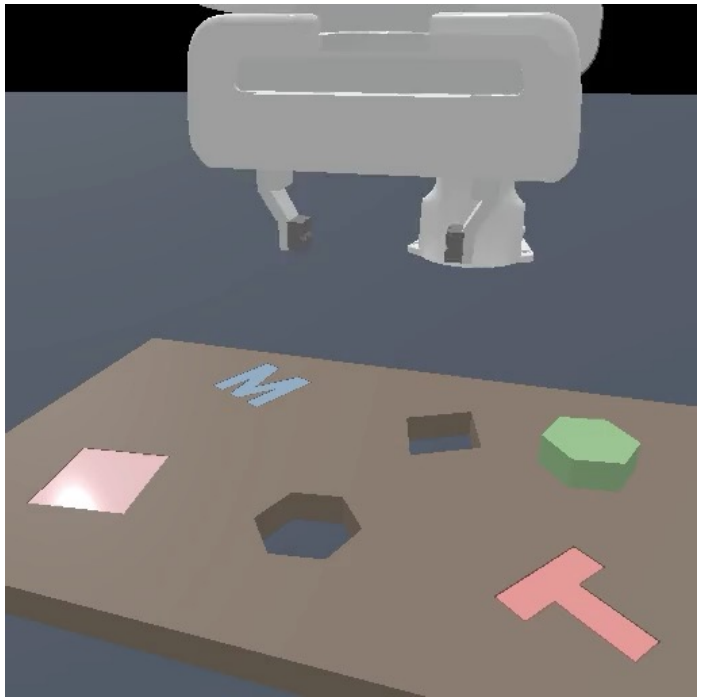


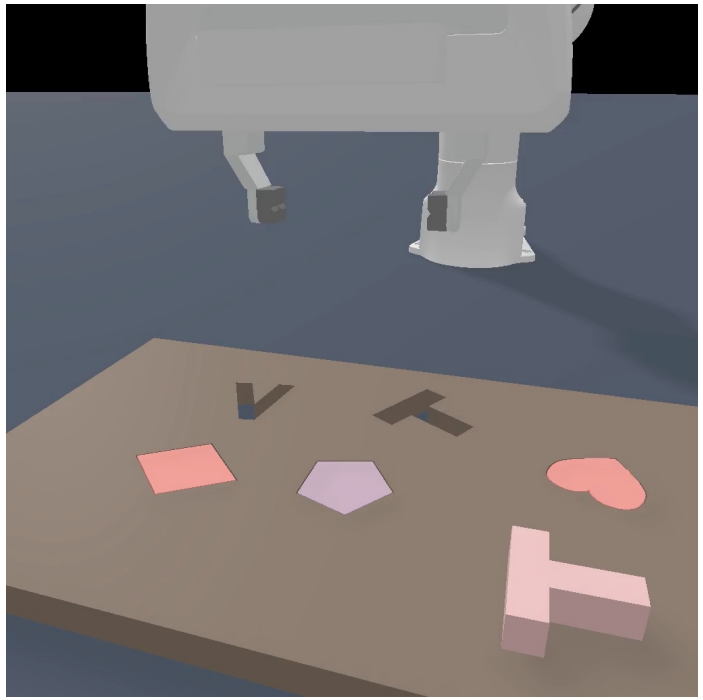
Failure: bad grasp point

Failure: small object

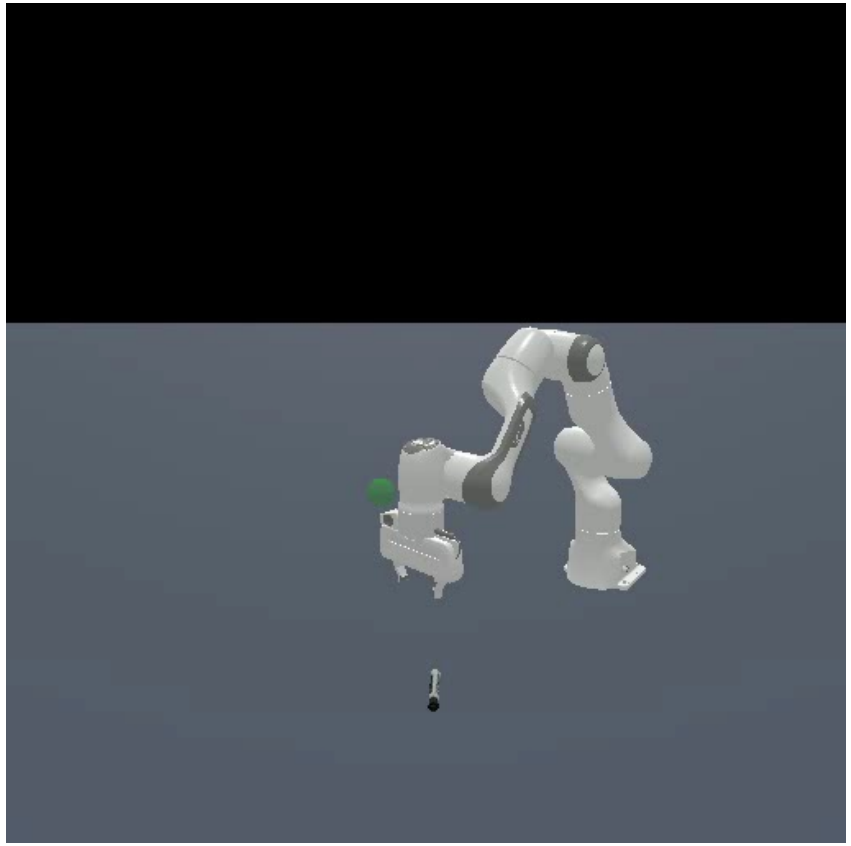
Example 2: Transporter Network

- The clearance is 0.8mm
- The pose accuracy (1cm/15deg) is 96%, while the success rate is 18%





Example 3: DAPG for PickSingleYCB



Success

Failure

DAPG

- Point cloud observations show better performance than RGBD
- Adding goals into visual inputs can be helpful
- Use point clouds in the end-effector frame is easier to learn

Obs. Mode	PickCube	StackCube	PickSingleYCB	PegInsSide	PlugCharger	AssemblingKits	TurnFaucet	AvoidObstacles
Point Cloud	0.94 ± 0.03	0.95 ± 0.02	0.51 ± 0.05	0.01 ± 0.01	0.01 ± 0.02	0.00 ± 0.00	0.04 ± 0.03	0.00 ± 0.00
RGBD	0.91 ± 0.05	0.87 ± 0.04	0.18 ± 0.07	0.01 ± 0.01	0.01 ± 0.01	0.00 ± 0.00	0.03 ± 0.03	0.00 ± 0.00

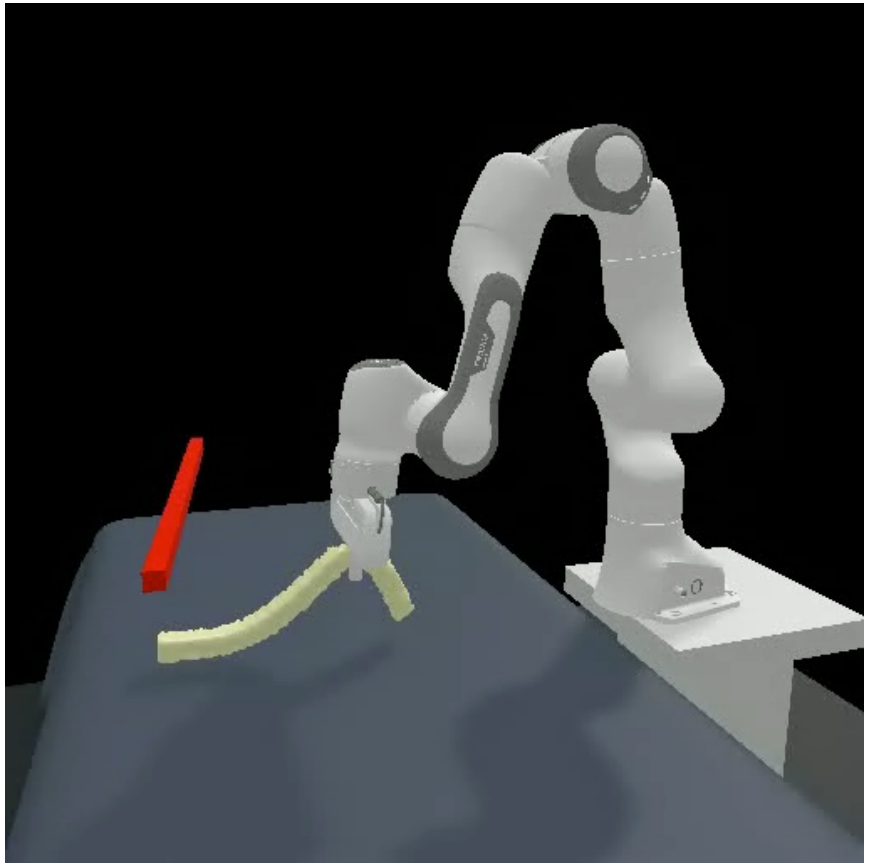
Table 3: Mean and standard deviation of success rates of DAPG+PPO on rigid-body tasks. Training budget is 25M time steps.

Example 4: Imitation Learning

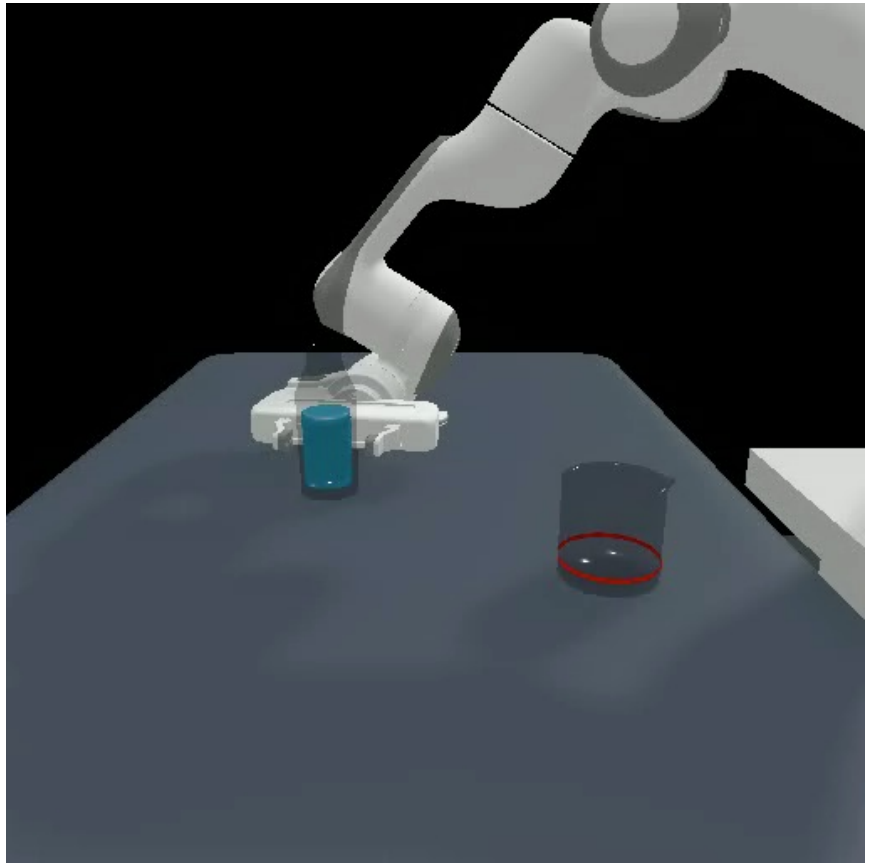
Excavate

Fill

Example 4: Imitation Learning

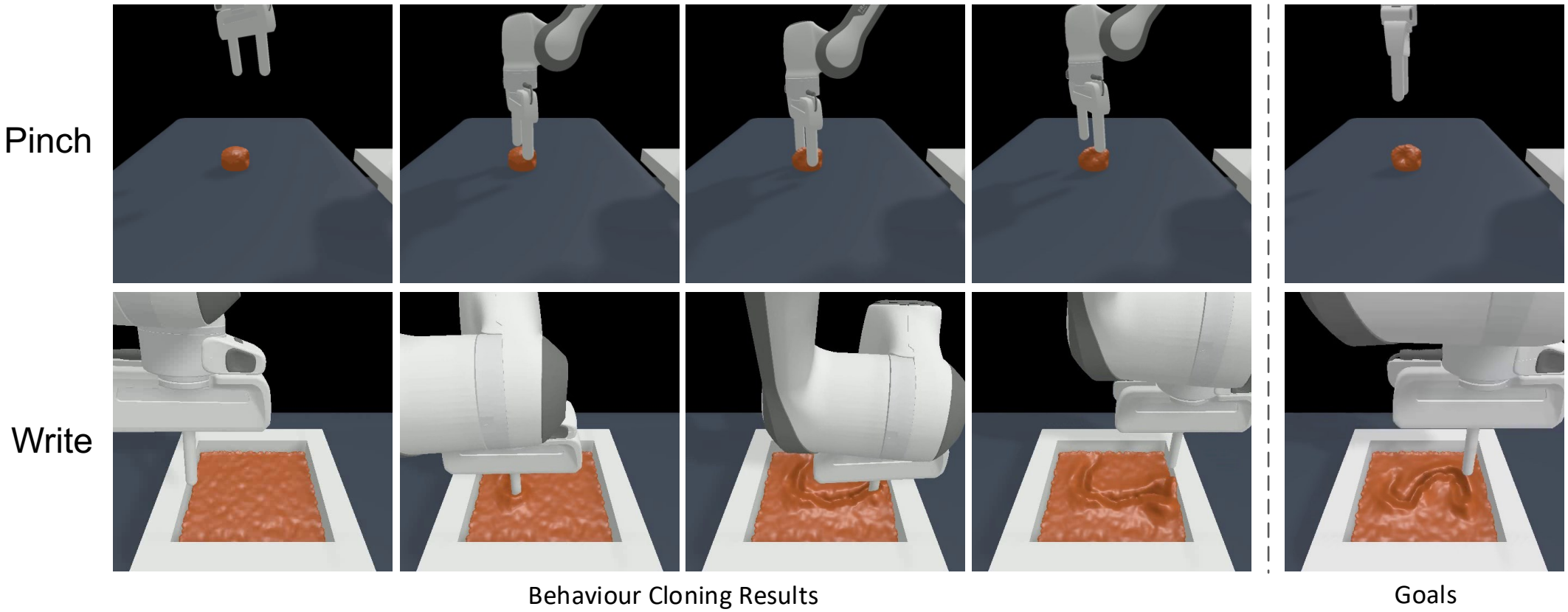


Hang

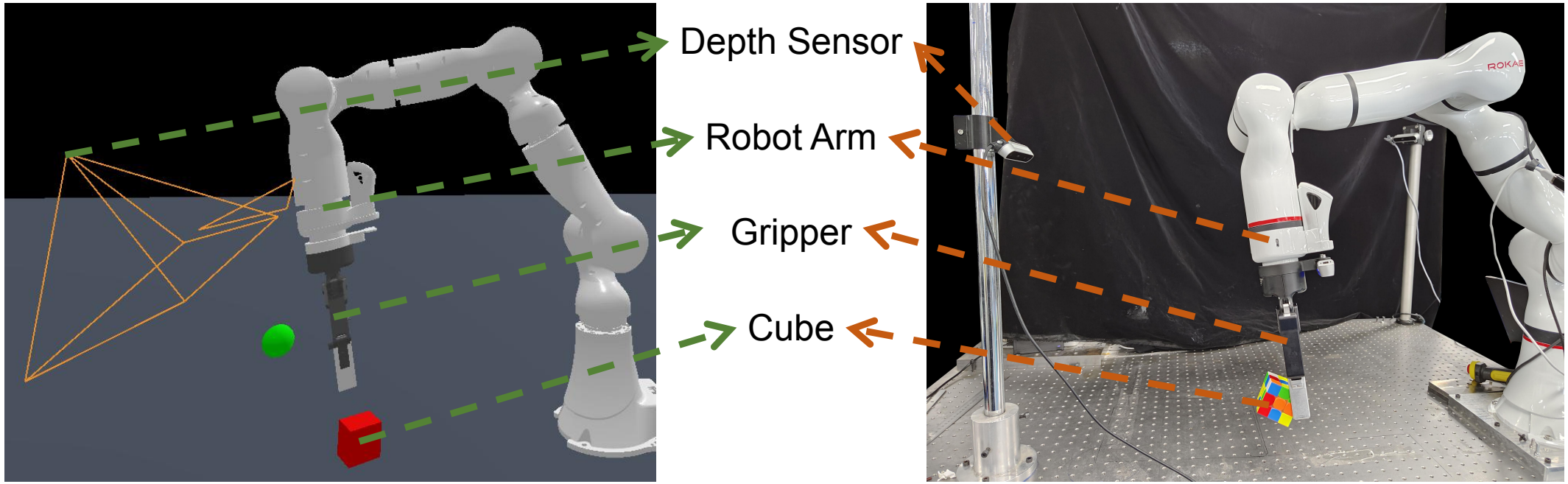


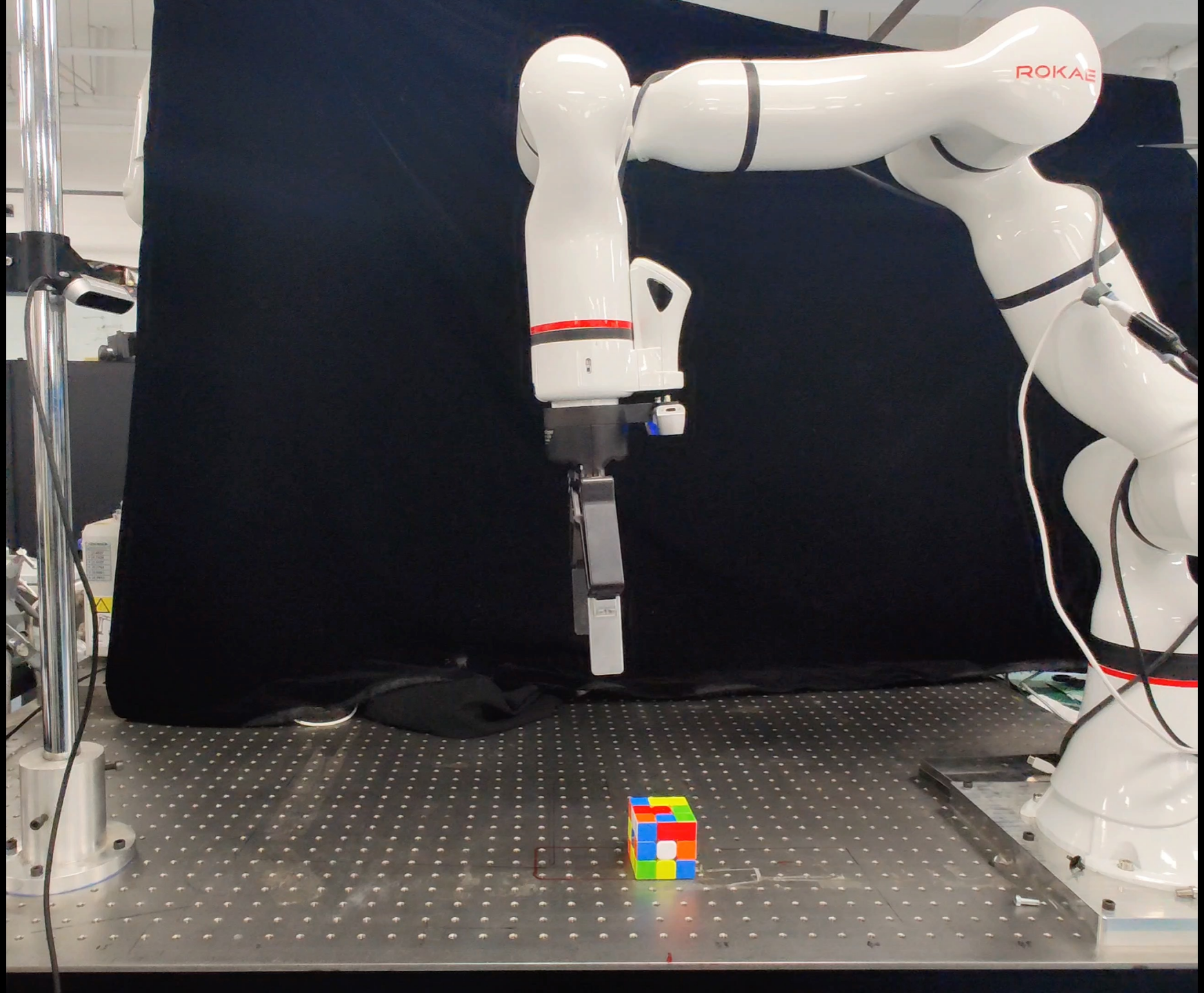
Pour

Example 4: Imitation Learning



Example 5: Sim2Real




ROKAE

Acknowledgement

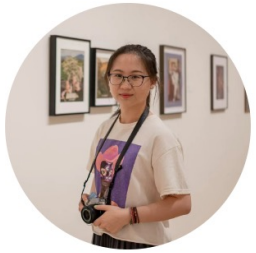

Jiayuan Gu
UC San Diego


Fanbo Xiang
UC San Diego


Rui Chen
Tsinghua University


Stone Tao
UC San Diego


Hao Su
UC San Diego


Xinyue Wei
UC San Diego


Yihe Tang
UC San Diego


Xiaodi Yuan
Tsinghua University


Zhiao Huang
UC San Diego


Xiqiang Liu
UC San Diego


Tongzhou Mu
UC San Diego


Zhan Ling
UC San Diego


Xuanlin Li
UC San Diego

Takeaway

- ManiSkill2 is a unified benchmark for learning generalizable manipulation skills with diverse assets and large-scale demonstrations

- After generalizable skills are learned, we can tackle more complicated, long-horizon tasks by chaining these skills

Multi-skill Mobile Manipulation for Object Rearrangement

Jiayuan Gu, Devendra Singh Chaplot, Hao Su, Jitendra Malik

UC San Diego, Meta AI, UC Berkeley

ICLR 2023

Long-horizon Tasks are Difficult

- Decompose the full task into subtasks
  - Task and Motion Planning (TAMP)
  - Hierarchical reinforcement learning (HRL)
- A skill is learned for each subtask
- At the core is **subtask formulation**

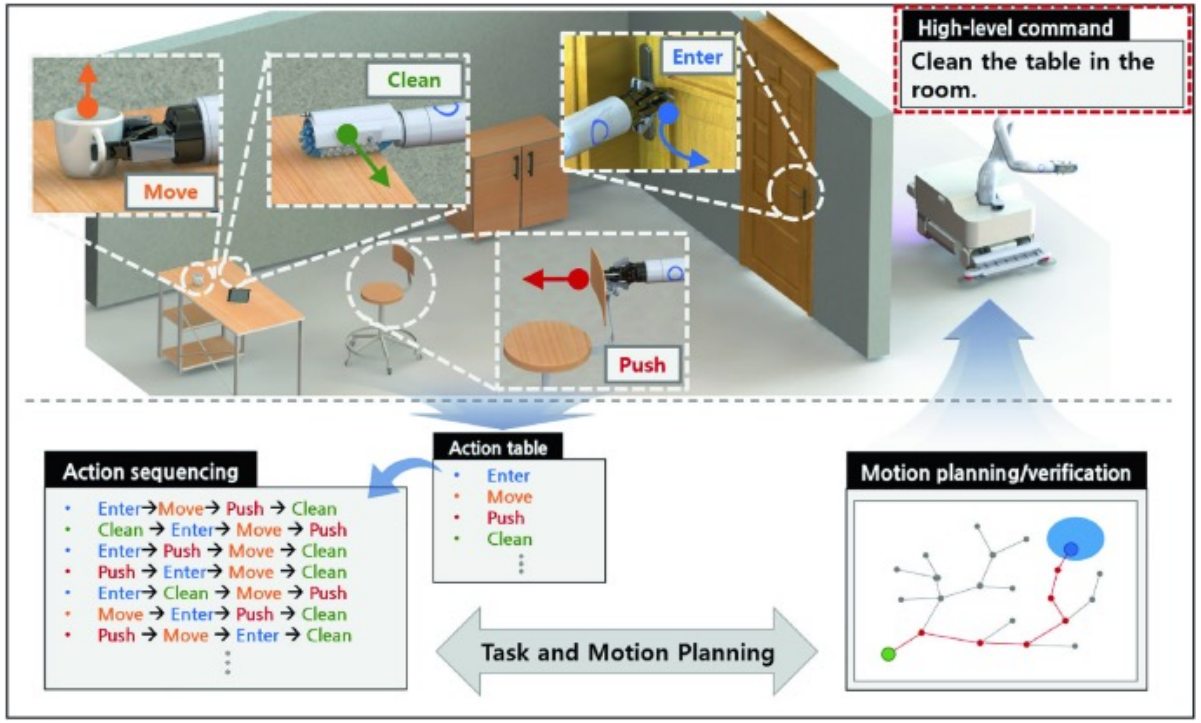
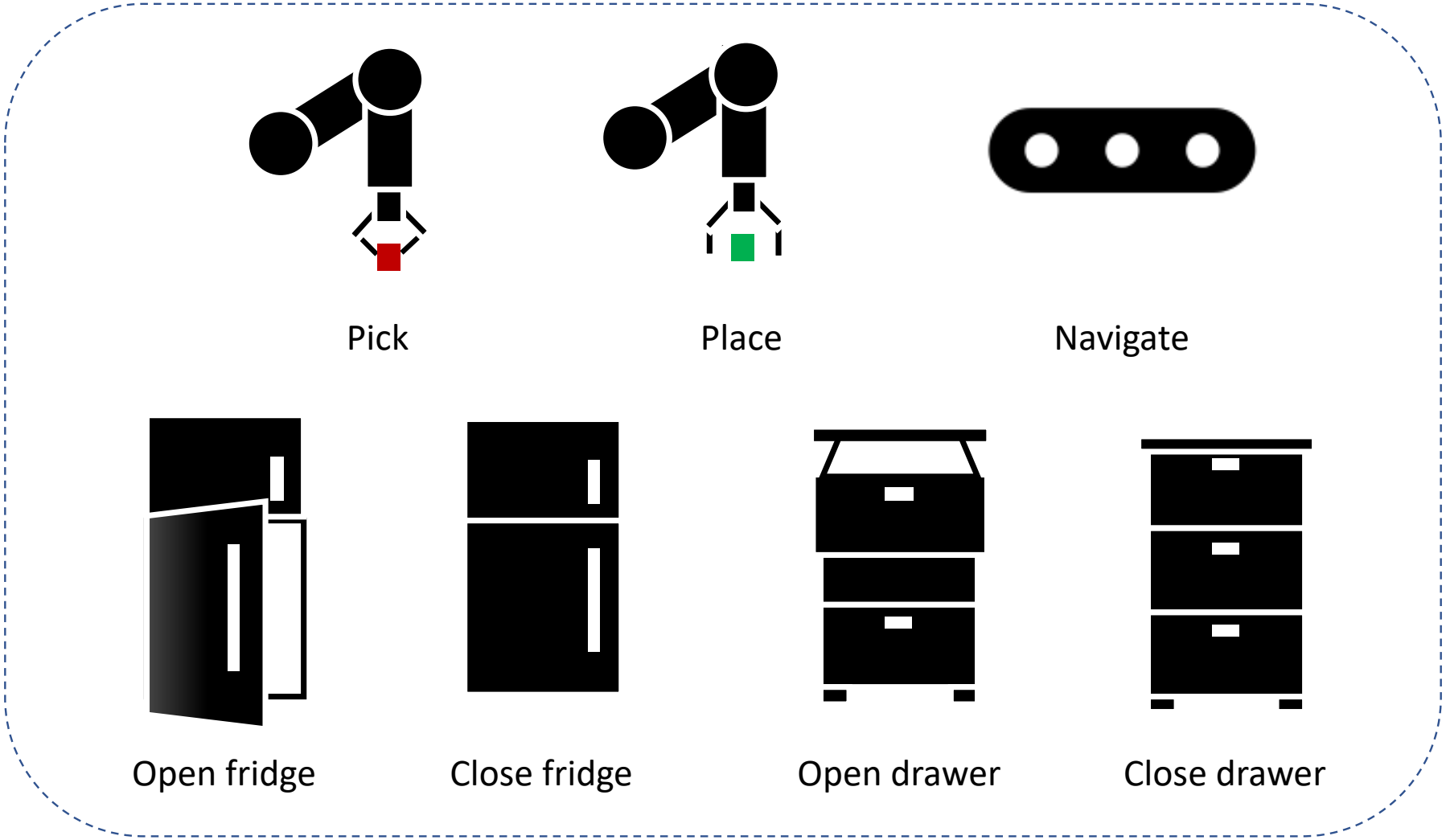
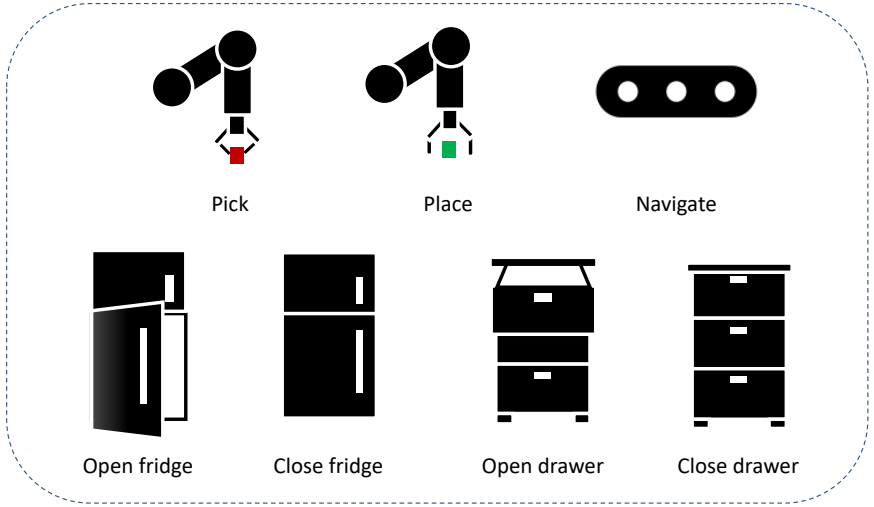


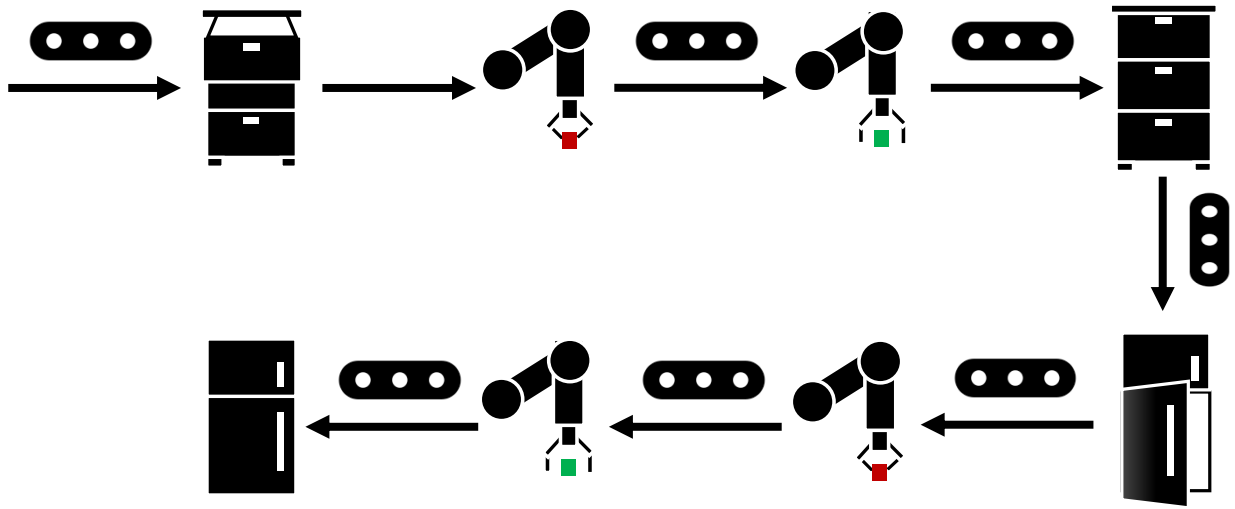
Figure from *Action Feasibility Learning with Cell-Based Multi-Object Representation for Task and Motion Planning*

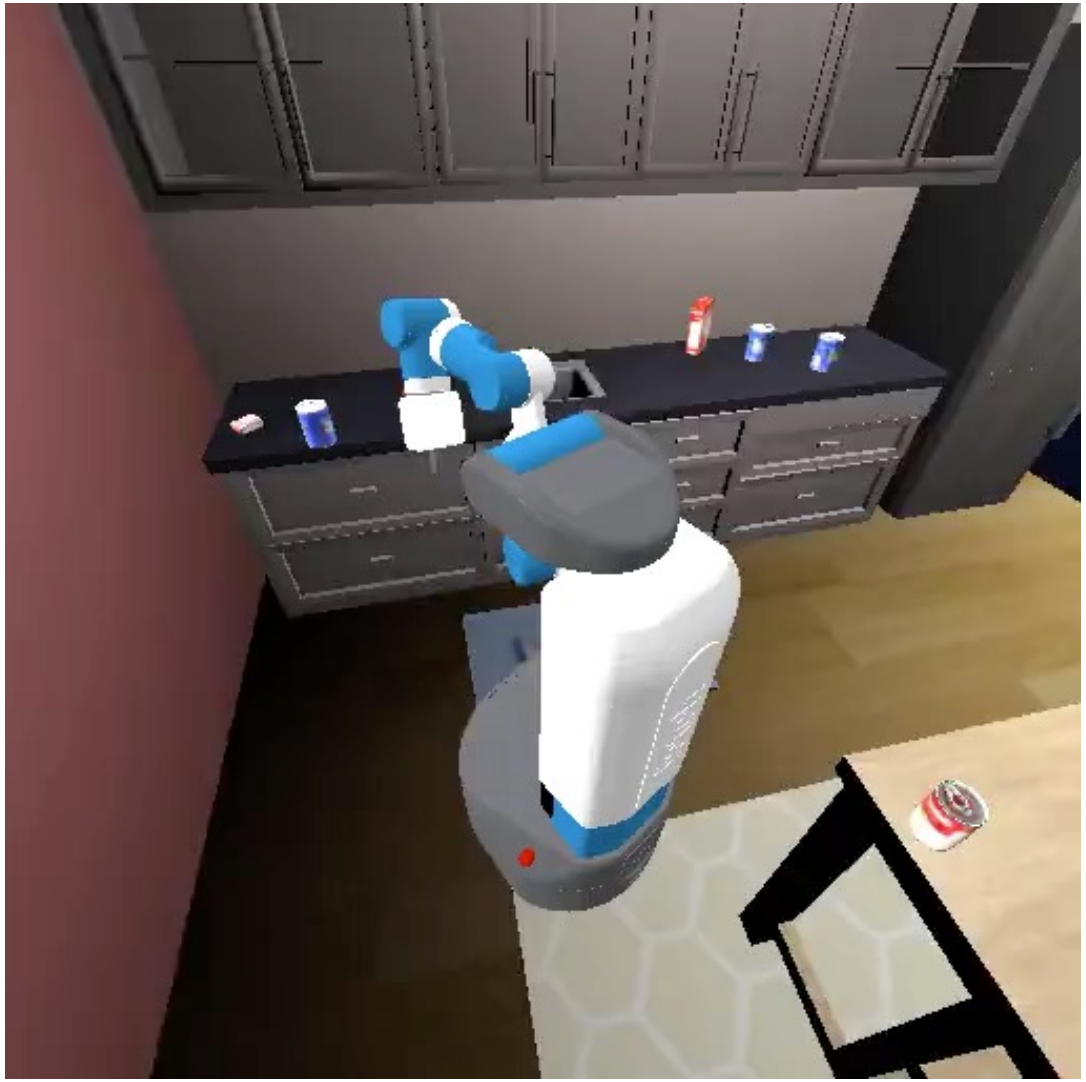
We train a set of RL skills to solve subtasks



The full task is solved by skill chaining







Subtask, Skill, and Skill Chaining

- Subtask: MDP associated with initial and terminal states
- Skill is learned (or scripted) for each subtask
- Skill chaining can be a criterion of subtask formulation, since a good subtask should lead to **a reusable and tractable skill that can be chained with other skills**

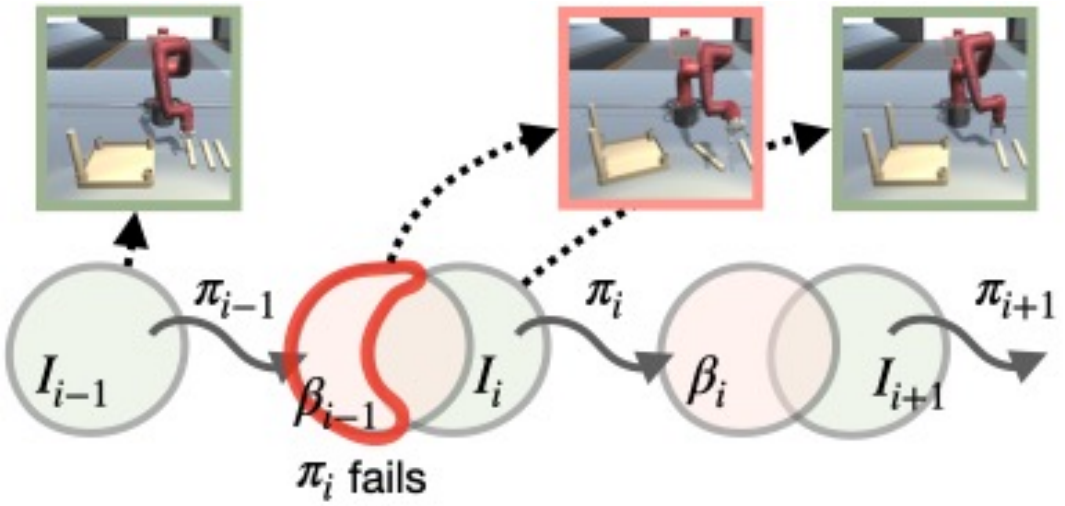
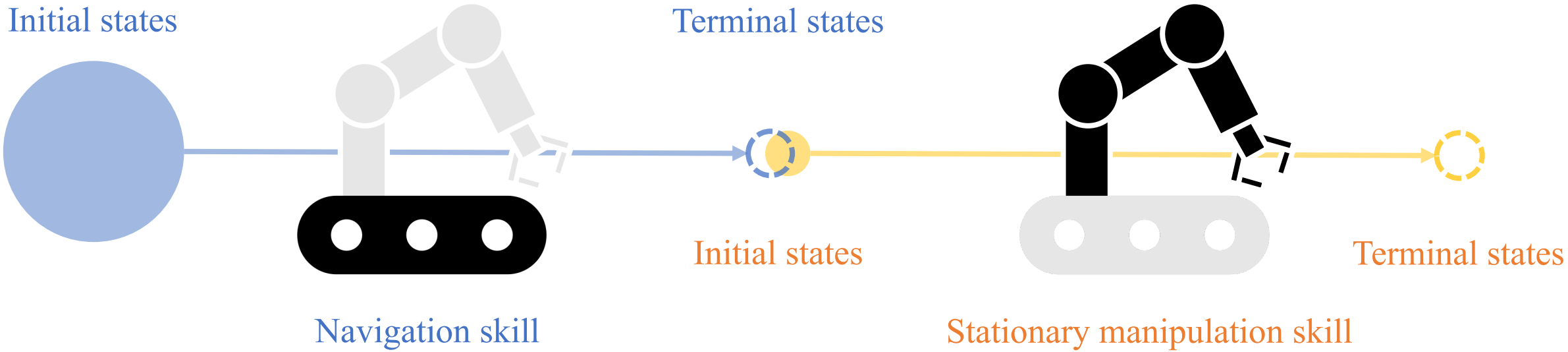


Figure from *Adversarial Skill Chaining for Long-Horizon Robot Manipulation via Terminal State Regularization*

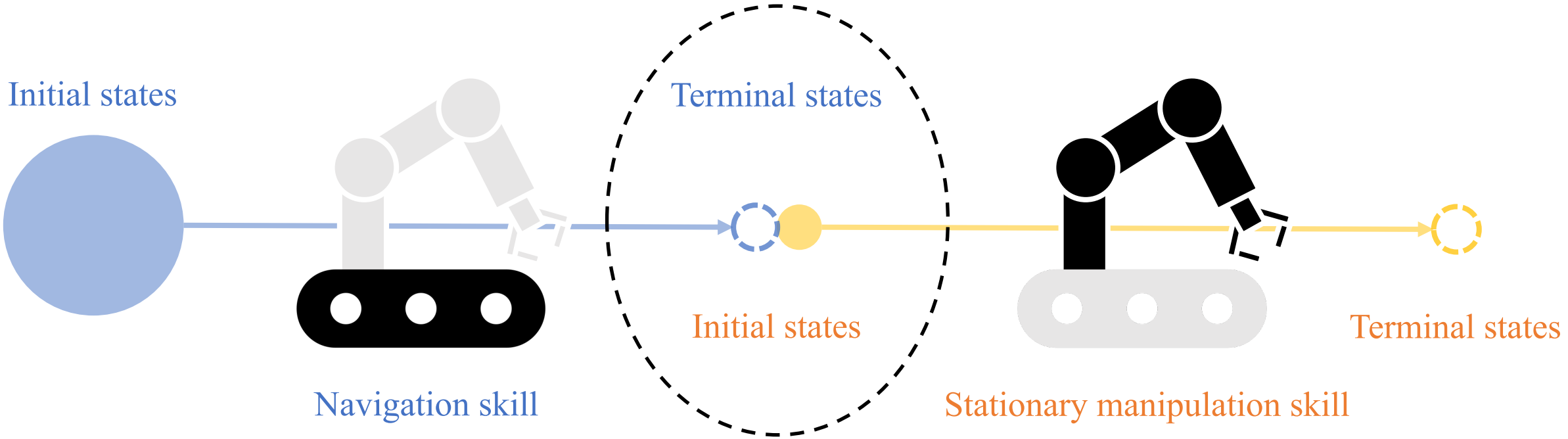
Principles to Formulate Skills

- achievability (be able to solve tasks)
- composability (be able to connect with other skills)
- reusability (least fine-tuning)

However, most prior works in RL and robotics separate the mobile platform and manipulator.



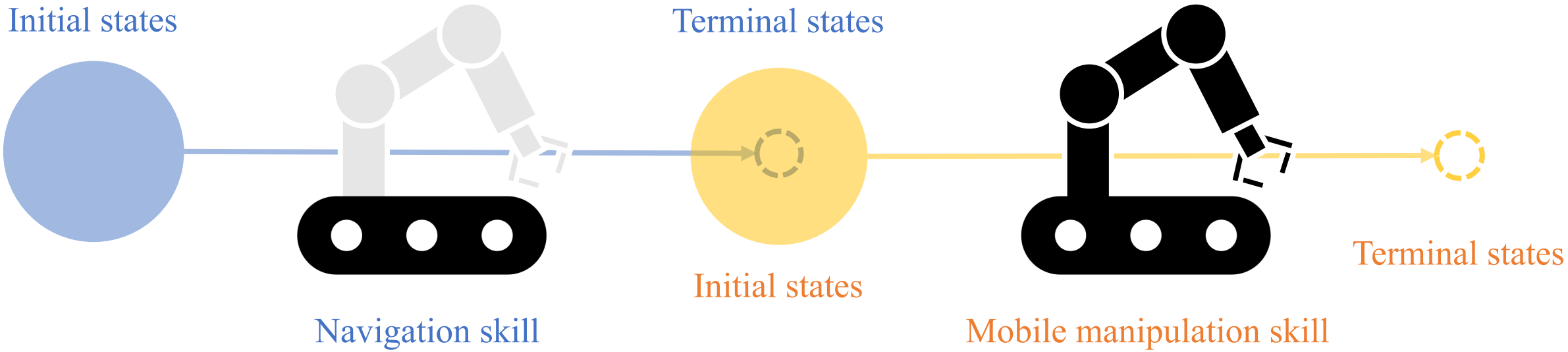
Prior methods chain the navigation skill and **stationary manipulation skills**



Problem: Suffer from compounding errors in skill chaining

Example: hand-off problem

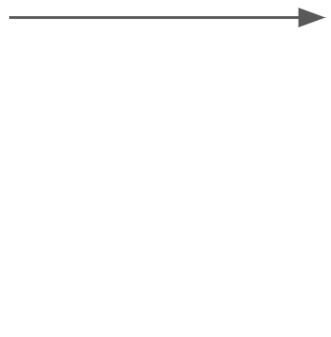
Fail to reach the target object, if the robot is not close enough to the object



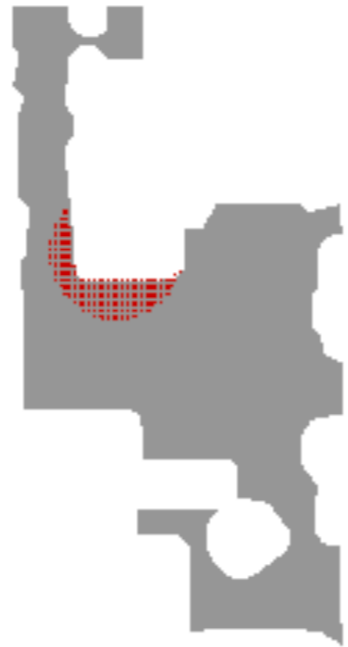
Solution: switch to mobile manipulation skills

Mobility can compensate for errors from the navigation skill

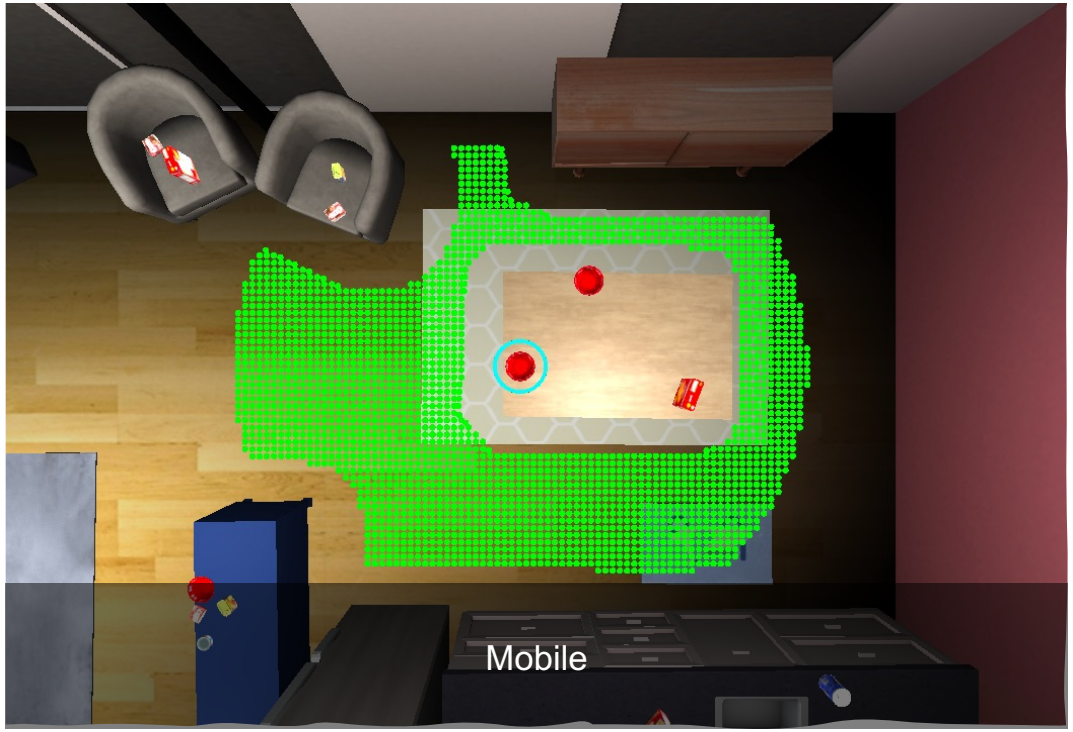
Discretization

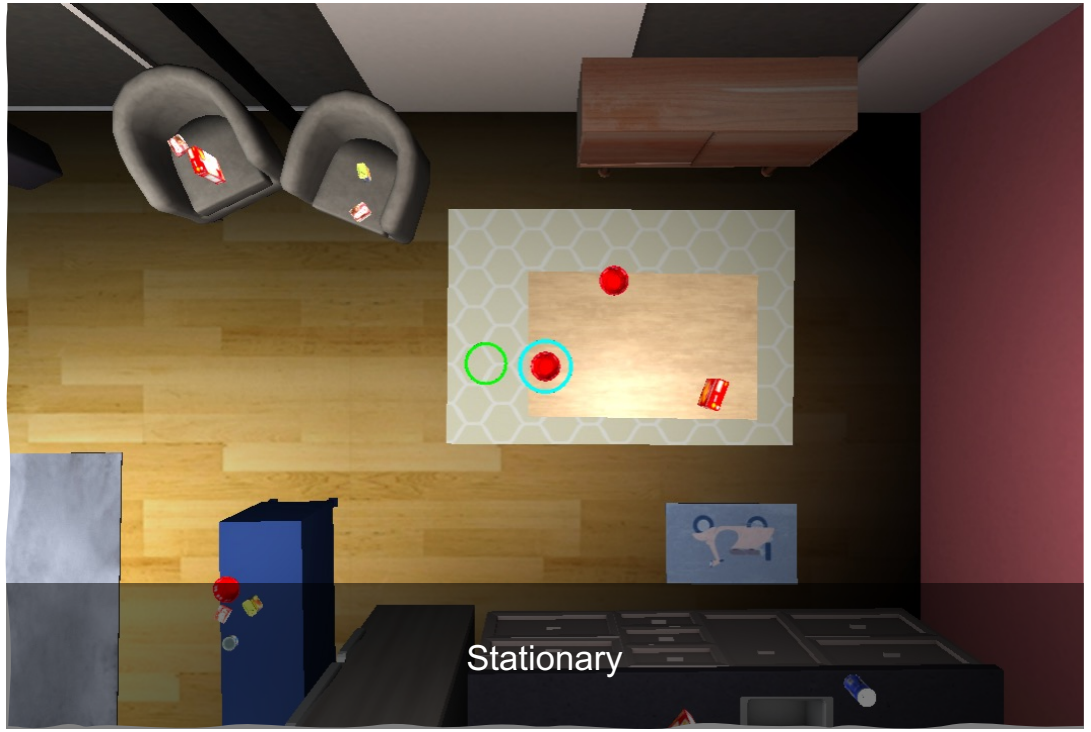


Restrict a region around the goal

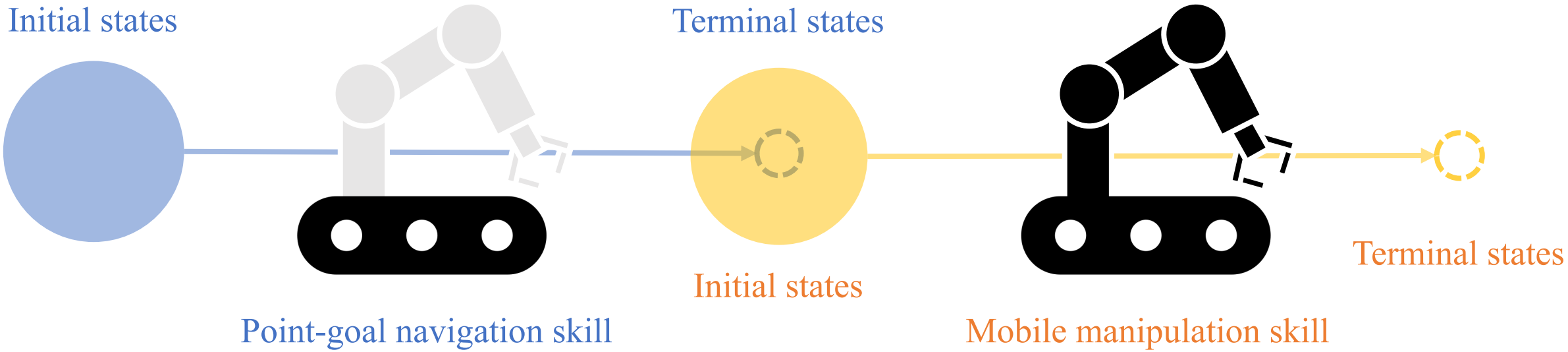


Post-process (at the largest island, snap to navigable points)

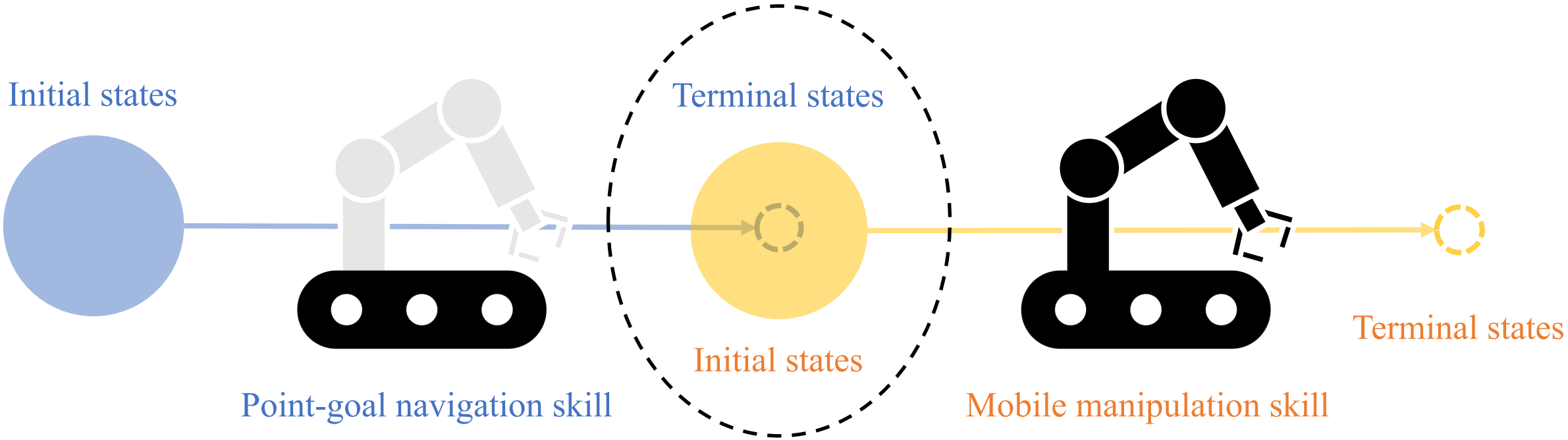

Mobile


Stationary

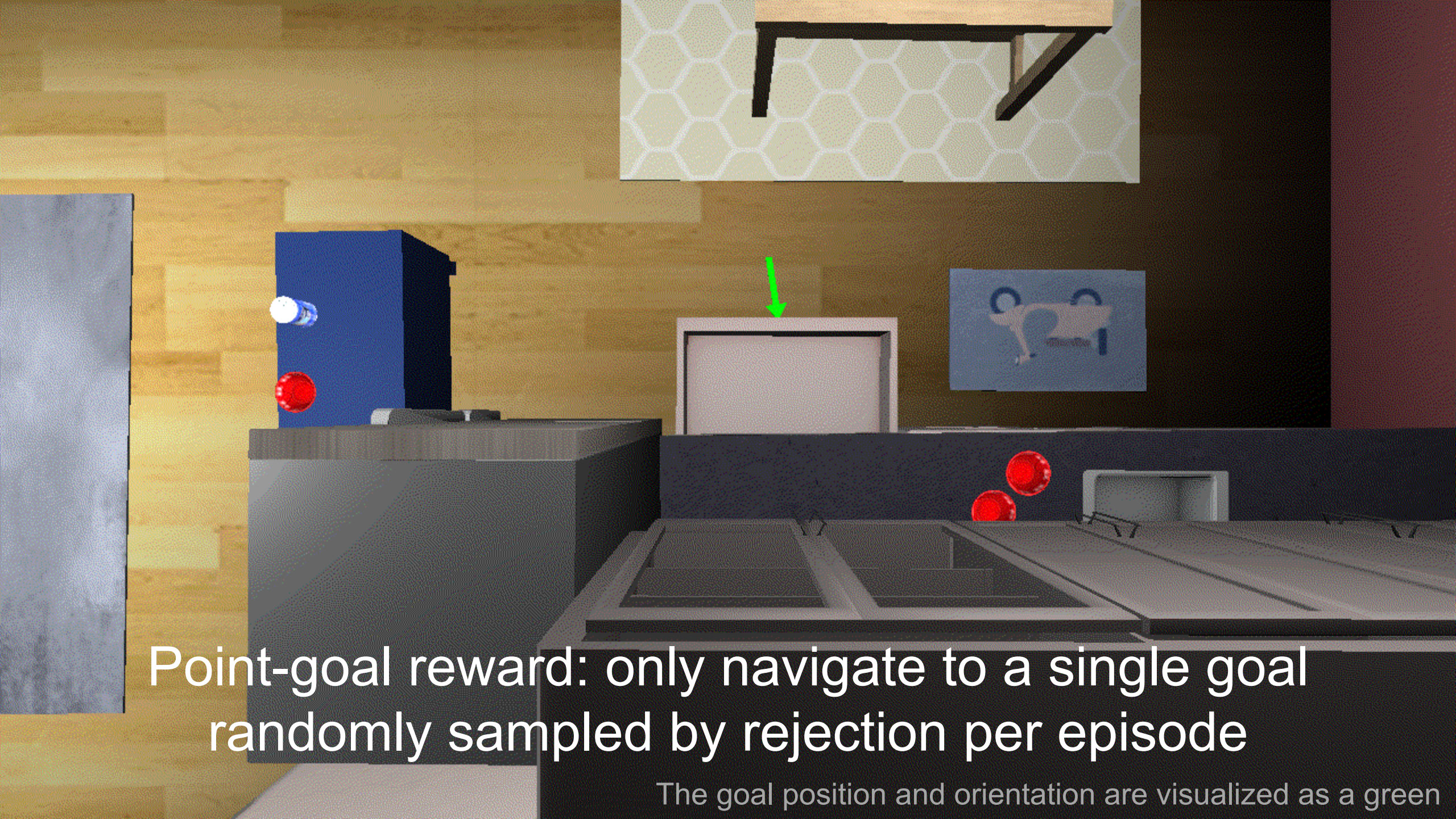
Thanks to the flexibility of mobile manipulation skills, we devise a region-goal navigation reward to facilitate learning navigation skills



Prior methods use point-goal navigation rewards



Problem: Ignore the existence of multiple good locations connecting navigation and manipulation


Point-goal reward: only navigate to a single goal randomly sampled by rejection per episode
The goal position and orientation are visualized as a green

Point-goal Navigation Reward

change of geodesic distance to the point goal

within a threshold of the point goal

slack penalty

$$r_t(s,a) = -\Delta_{geo} - \lambda_{ang}\Delta_{ang}I_{[d_t^{geo} \leq \tilde{D}]} + \lambda_{succ}I_{[d_t^{geo} \leq D \wedge d_t^{ang} \leq \Theta]} - r_{slack}$$

change of angular distance to the desired orientation

Indicator of success

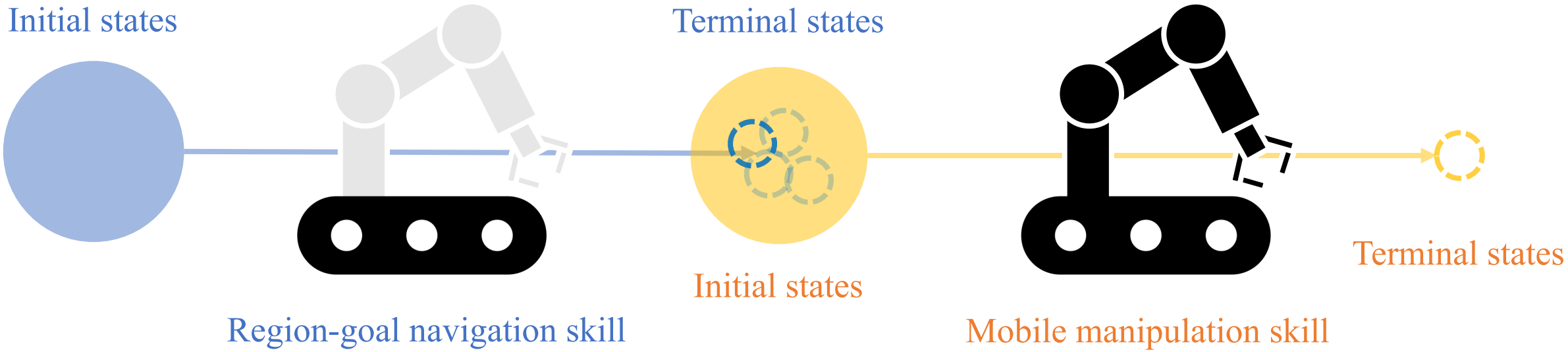
Point-goal Navigation Reward

point goal can be random per episode due to rejection sampling

$$r_t(s,a) = -\Delta_{geo} - \lambda_{ang}\Delta_{ang} I_{[d_t^{geo} \leq \tilde{D}]} + \lambda_{succ} I_{[d_t^{geo} \leq D \wedge d_t^{ang} \leq \Theta]} - r_{slack}$$

encourage entering the region by backing onto the target

The point-goal navigation reward can lead to poor convergence and generalizability due to goal ambiguity



Solution: a region-goal navigation reward to encourage the robot to reach any of feasible goals in a region


Region-goal reward: navigate to any feasible goal within the region
The feasible goal positions are visualized as green points

Region-goal Navigation Reward

change of geodesic distance to a region of goals

Add a collision penalty to filter infeasible goals

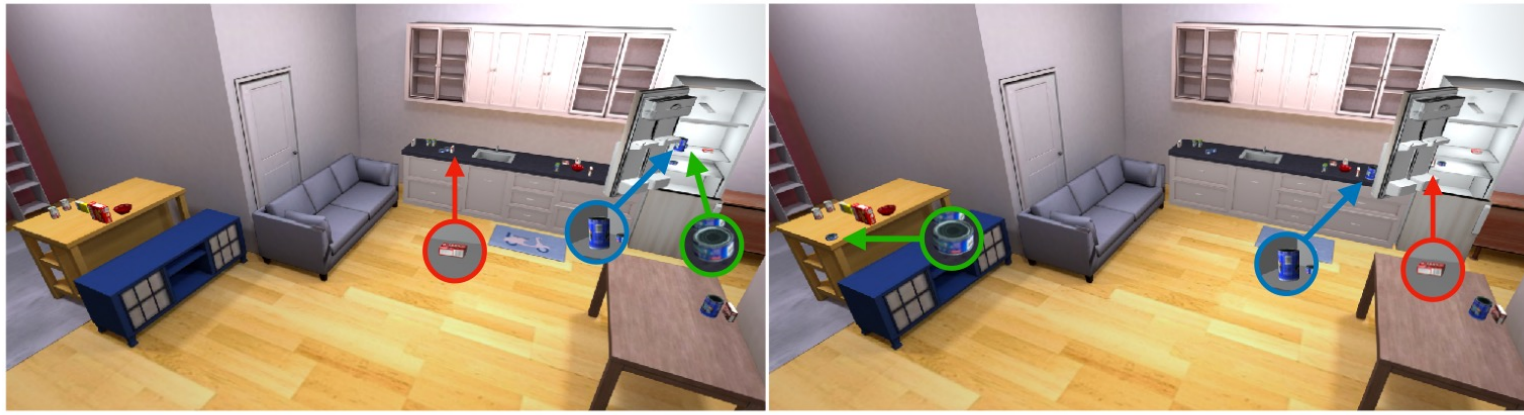
$$r_t(s,a) = -\Delta_{geo} + \lambda_{succ} I_{[d_t^{geo} \leq D \wedge d_t^{ang} \leq \Theta]} - r_{col} - r_{slack}$$

only check angular distance in success indicator

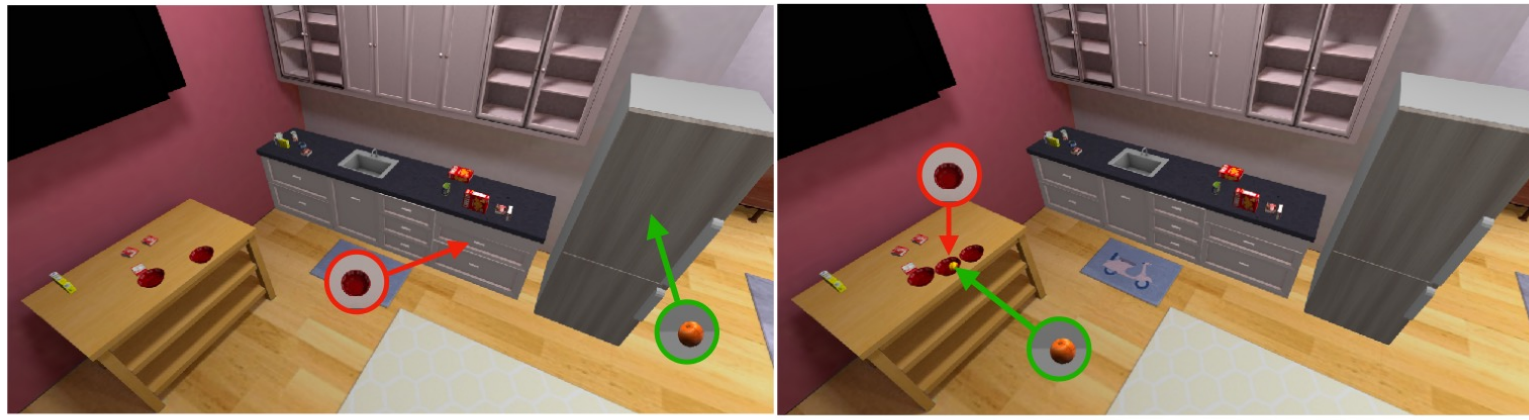
We can reuse candidates for base positions as the region of goals!

Home Assistant Benchmark (HAB)

(a) TidyHouse



(b) PrepareGroceries



(c) SetTable

ReplicaCAD



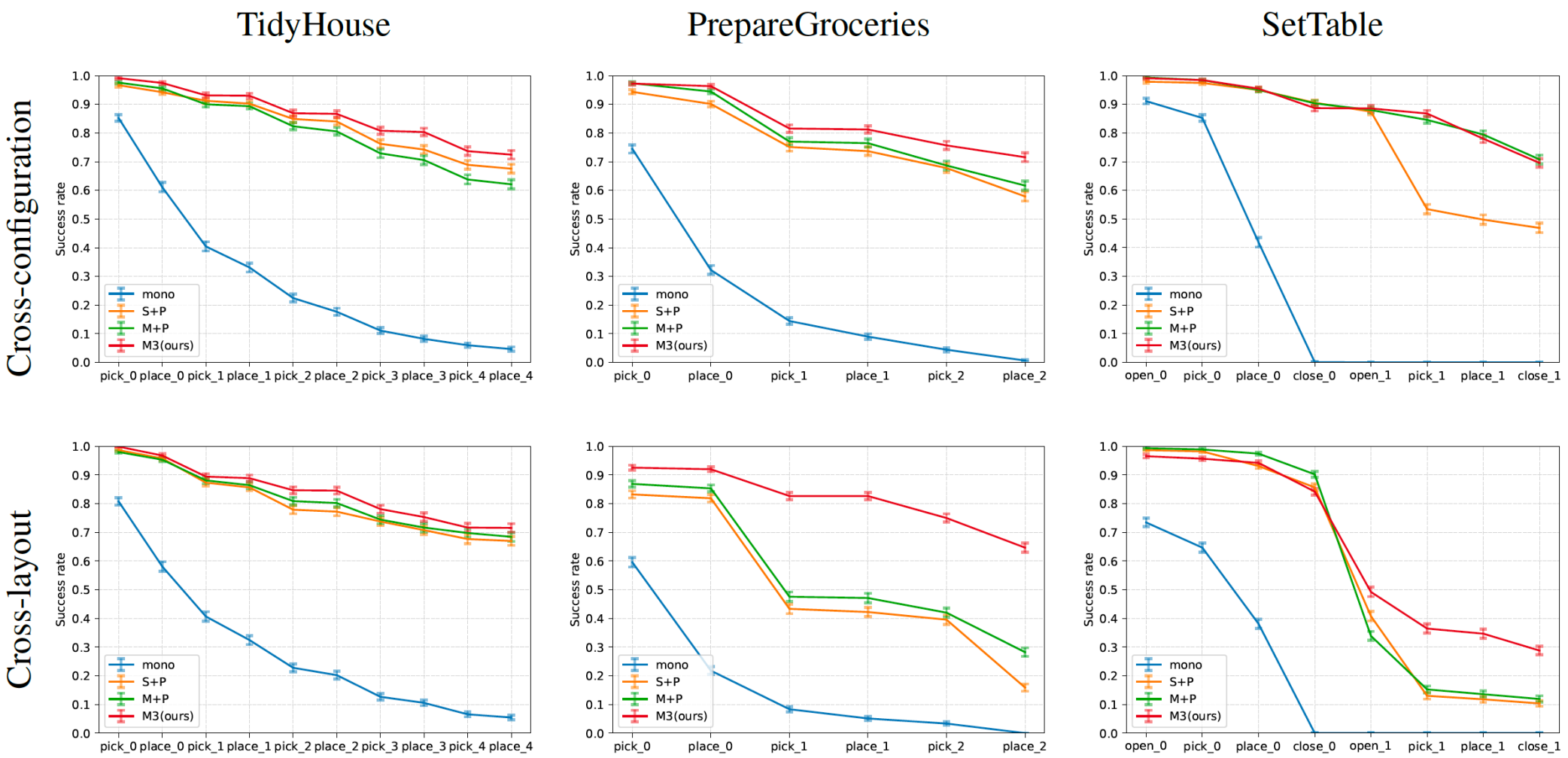


64 training layouts
5000 training episodes

20 val layouts similar to train ones
100 val episodes
(cross-configuration)

20 held-out layout
100 test episodes
(cross-layout)

Quantitative Results



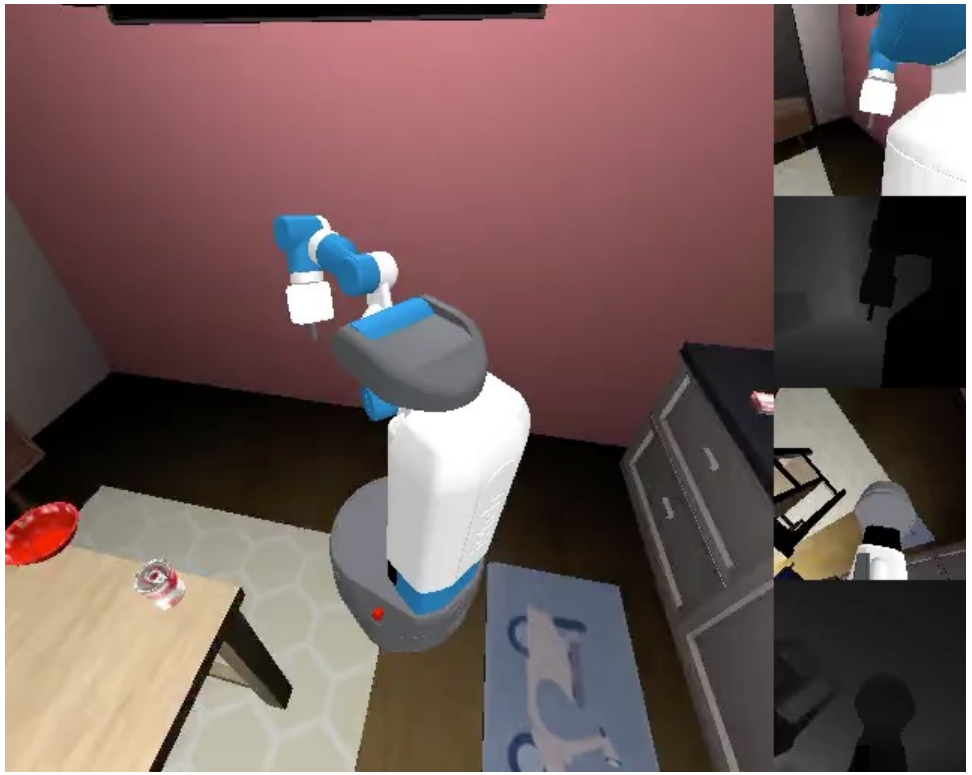
Qualitative Examples

Baseline1
Stationary Manipulation Skill
and
Point-goal Navigation skill

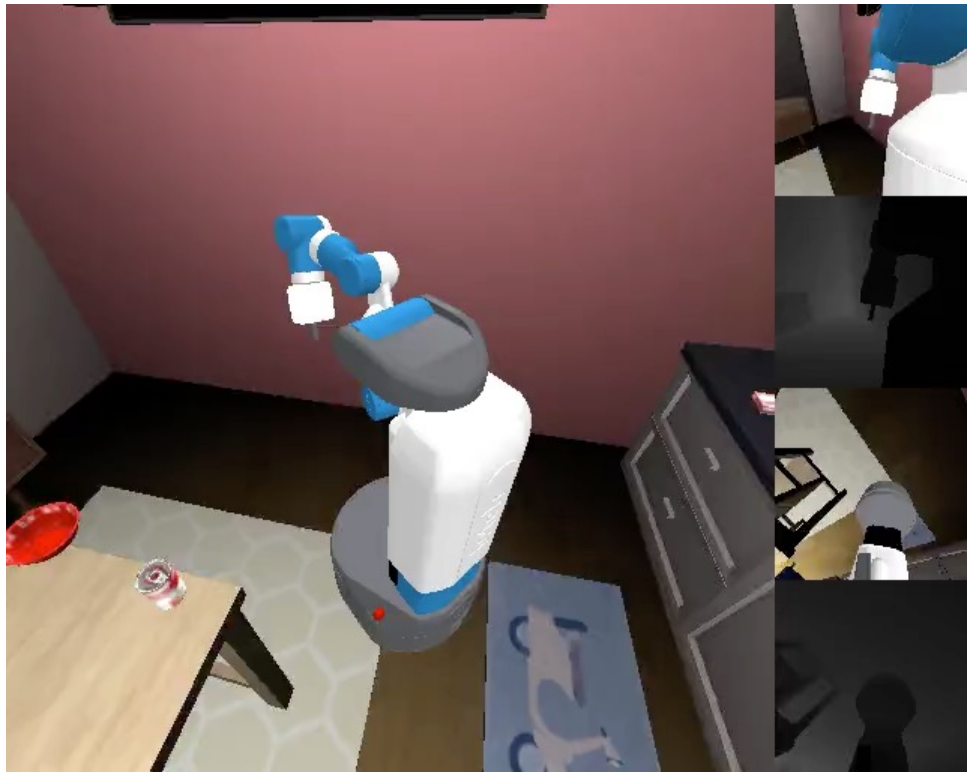
Baseline2
Mobile Manipulation Skill
and
Point-goal Navigation skill

Ours
Mobile Manipulation Skill
and
Region-goal Navigation skill

Better Manipulation Location



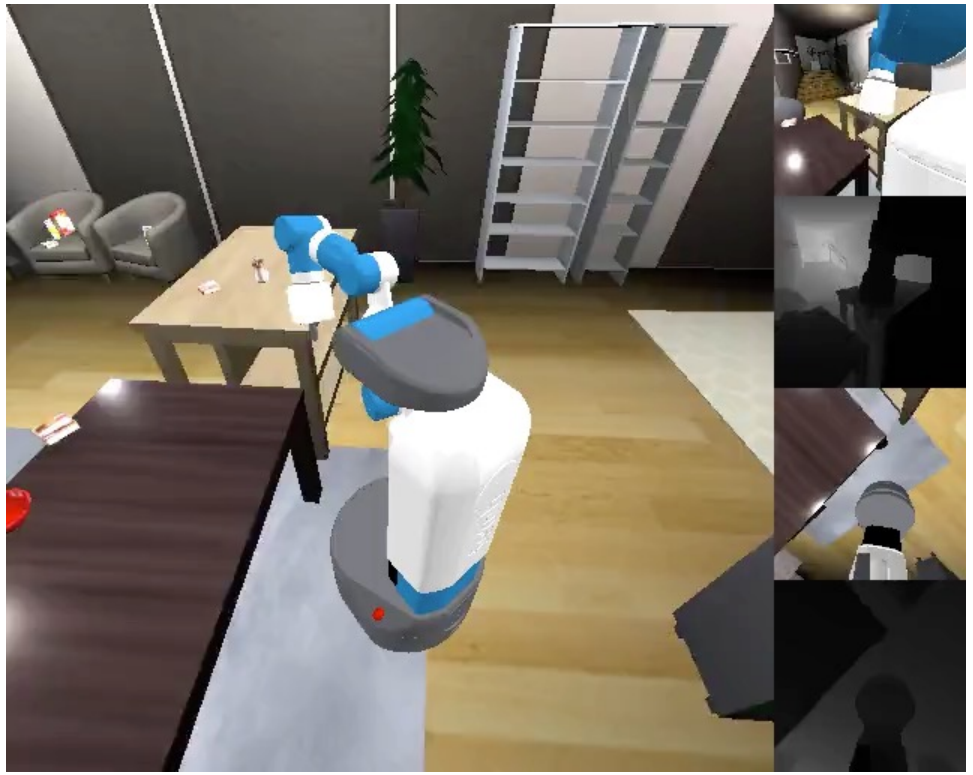
Baseline
Stationary Manipulation Skill



Ours
Mobile Manipulation Skill

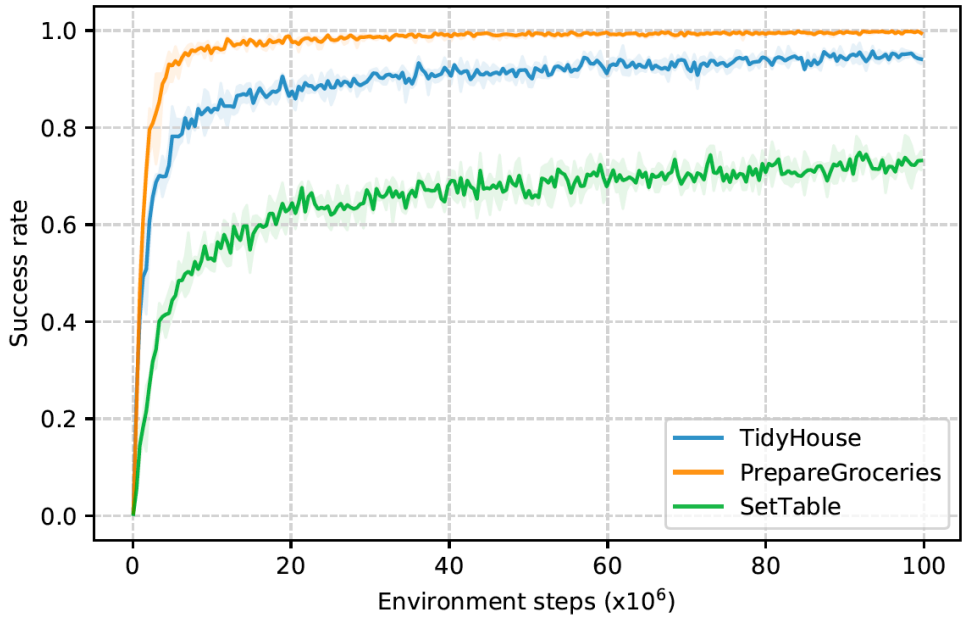
Collision-aware Navigation

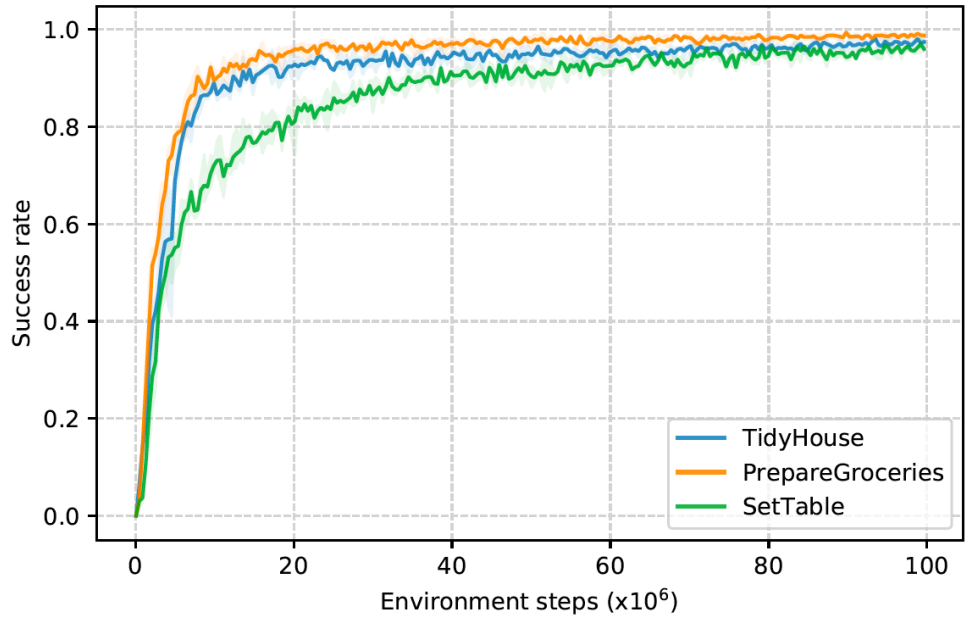
Baseline
Point-goal Navigation Skill



Ours
Collision-aware Region-goal Navigation Skill

Better Navigation Achievability





(k) Navigation (point-goal) (l) Navigation (region-goal)

Takeaway

- stationary manipulation -> mobile manipulation
  - compensate for imperfect navigation
  - find a better location for manipulation
  - reduce heuristics to design feasible initial states
- point-goal navigation reward -> region-goal navigation reward
  - less memorization, better generalization
  - reduce heuristics to design feasible point goals

Acknowledgement

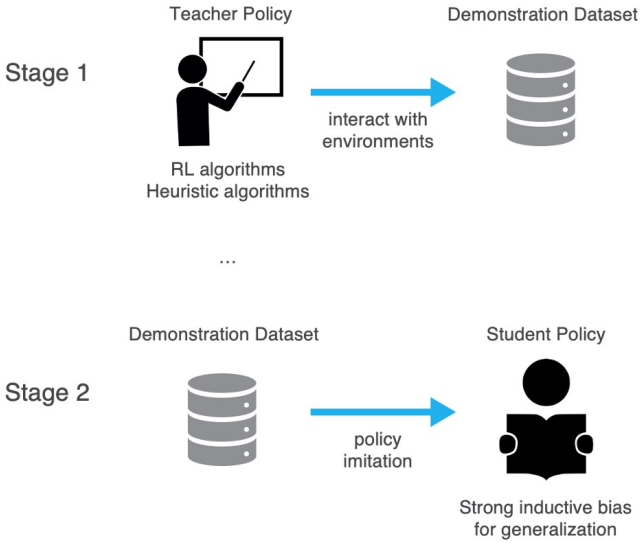
Jiayuan Gu
UC San Diego

Devendra Singh Chaplot
Meta AI

Hao Su
UC San Diego

Jitendra Malik
UC Berkeley

Conclusion: Create Data and Learn Skills



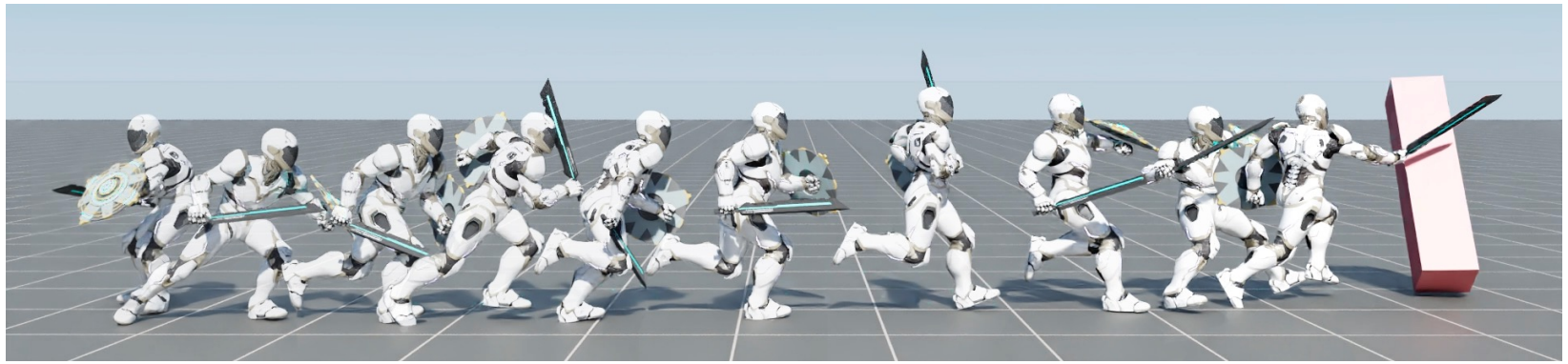
Acquire generalizable skills from offline data (policy refactorization)

Build environments to acquire data (ManiSKill2)

Design and learn generalizable visuomotor skills (M3)

Future Application: Character Motion

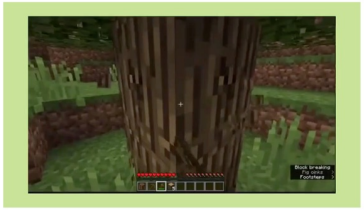
- Learning natural behaviors for physically simulated characters
  - Make use of large-scale motion datasets
  - Formulate as RL and learn with intrinsic rewards



ASE: Large-Scale Reusable Adversarial Skill Embeddings for Physically Simulated Characters

Future Application: Game AI

- Evolve game AI from large-scale data
  - Collect data from users, internet videos, forums
  - Bootstrap game AI by unsupervised learning, inverse RL, or others



Video clip 1

Video clip 2

Video clip 3

… but I'm gonna go around gathering a little bit more **wood** from these **trees**.

Aligned text 1

… I'm also going to gather a little bit of **stone** from the side of this little hill here.

Aligned text 2

As I raised my axe in front of this **pig**, there's only one thing you know is gonna happen…

Aligned text 3

Simulator

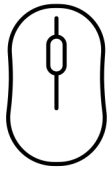
MINEDOJO: Building Open-Ended Embodied Agents with Internet-Scale Knowledge

Q&A

What can affect your RL algorithms?

Dense reward

Observations



Action space

Training scheme

Dense Reward

- Which reward can be better for PickCube?

$$r_t = [1 - \tanh(d_t(tcp, cube))] + [1 - \tanh\big(d_t(gpal, cube)\big)]$$

$$r_t = -d_t(tcp, cube) - d_t(goal, cube)$$

$$r_t = [d_{t-1}(tcp, cube) - d_t(tcp, cube)] + [d_{t-1}(goal, cube) - d_t(goal, cube)]$$

Observations

- Do not miss important information
- If your reward function includes multiple stages, it is better to include some signals in observations
  - For example, whether the cube is grasped in PickCube
    - a flag using GT contact information (not accessible in the real world)
    - leveraging previous action and current position of the gripper

Action Space

- End-effector space (a.k.a task space) can be easier than joint space for pick-and-place tasks

Training Scheme

- Is the task episodic or continuous?
  - episodic: the episode has a finite length.
  - continuous: the episode is infinite, and can be truncated for learning
- Do we terminate the episode if the task succeeds?