

The logo for Carnegie Mellon University, featuring a dark blue background with a grid of colorful lines (red, green, yellow, blue) forming a diamond pattern.

**Carnegie
Mellon
University**

SPIN: distilling Skill-RRT for long-horizon prehensile and non-prehensile manipulation

Feb 4, 2026

Mukai (Tom Notch) Yu

What are the tasks

From planning to policy: distilling Skill-RRT for long-horizon prehensile and non-prehensile manipulation



From planning to policy: distilling Skill-RRT for long-horizon prehensile and non-prehensile manipulation





From planning to policy: distilling Skill-RRT for long-horizon prehensile and non-prehensile manipulation



From planning to policy: distilling Skill-RRT for long-horizon prehensile and non-prehensile manipulation



What is being proposed

- What's prehensile (**P**) and non-prehensile (**NP**) ?
 - Prehensile (**P**) : grab/grasp 
 - Non-prehensile (**NP**) : push/press with fist 
- Framework: skill policies + RRT — to solve **long-horizon PNP**
- Planning trick:
 - i. Assume *teleportation* between skill end/start states
 - ii. Train connector policies to fill in the gap later
- Data generation
 - Chain skill policies via Skill-RRT
 - Treat resulting rollouts as “expert” demonstrations
- Distill “expert” data into 1 diffusion policy for 1 particular task
- **Not** about skill learning/discovery
 - All skill policies are human **pre-defined** and **pretrained with RL**

Why it is proposed

- Long-horizon PNP manipulation is hard (OOD definition):
 - Varying object/robot **initial states**
 - Varying object **goal states**
 - Compounding error
- Proposed approach:
 - Divide and conquer — divide into subtasks, conquered by skill policies
 - Plan in skill-level to handle long-horizon
 - Rely on skill-policies for local execution



Related Works

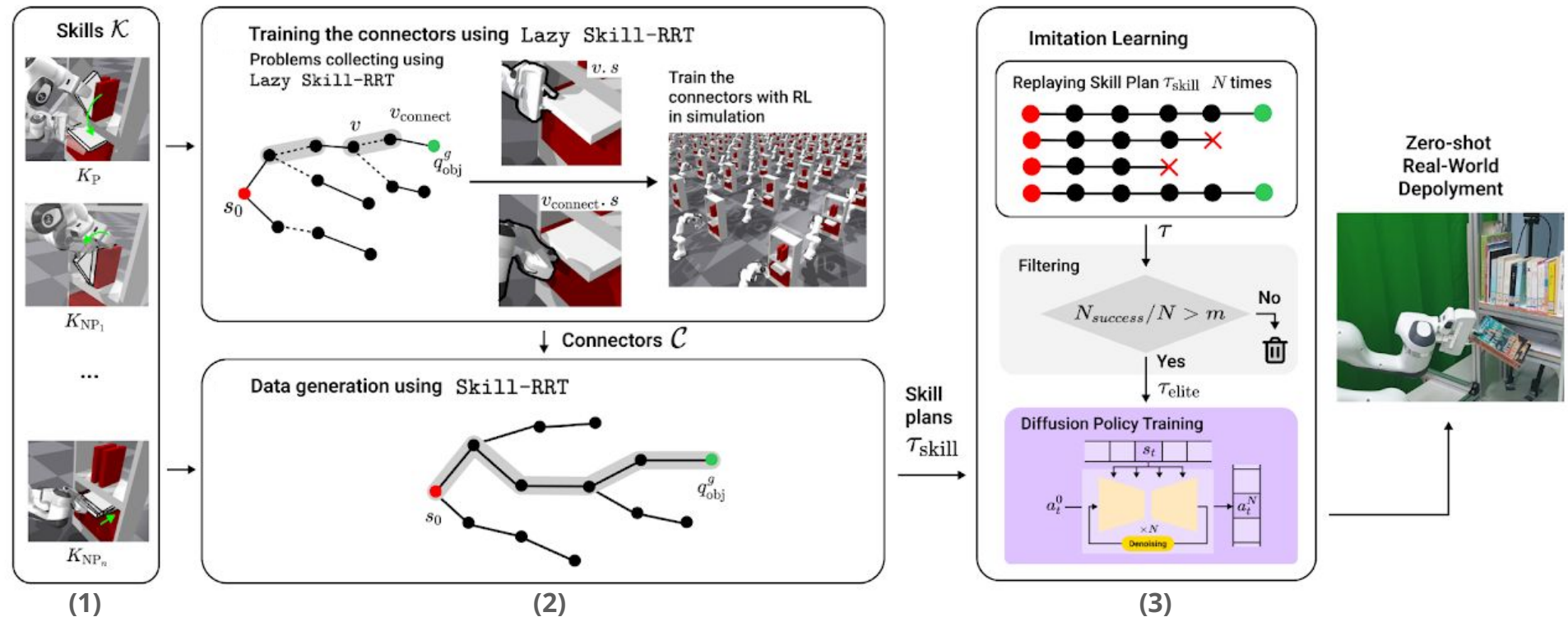
- Hierarchical Reinforcement Learning
 - Multi-level RL for dispatch skills/low-level actions
 - Learns hierarchy end-to-end or with predefined skills
- Parametrized Action MDPs (PAMDPs)
 - Discrete actions with continuous parameters
 - Used for manipulation primitives — push, grasp with target pose
- Task and Motion Planning (TAMP)
 - Planning over action primitives with continuous feasibility checks
 - Sampling-based planners for long-horizon tasks
- Planning-to-Policy Distillation
 - Generate demonstrations via planning
 - Distill into reactive policies for faster execution

Assumptions

- Skills are human pre-defined
 - Skill policies are pre-trained with RL
 - **Skill Card**: Skill policies have known:
 - capabilities/effects
 - prerequisites/applicability
 - termination conditions
- Environment and object are fixed for a particular task

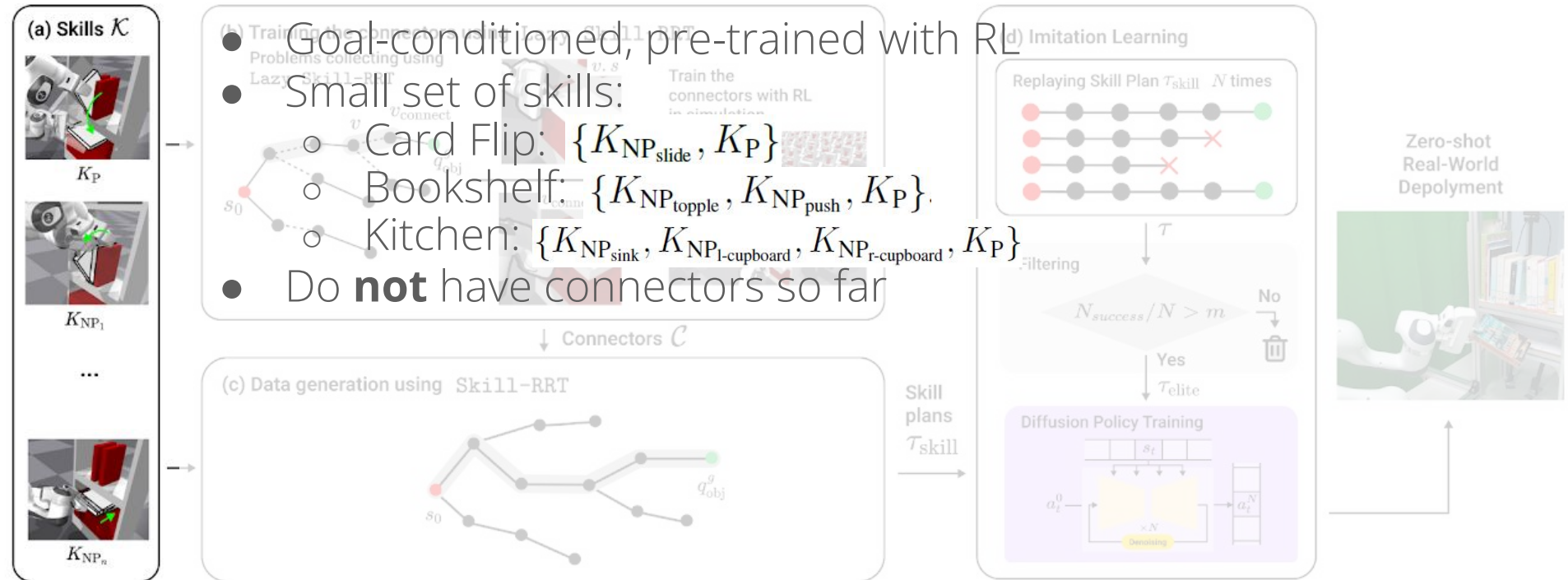


How it works

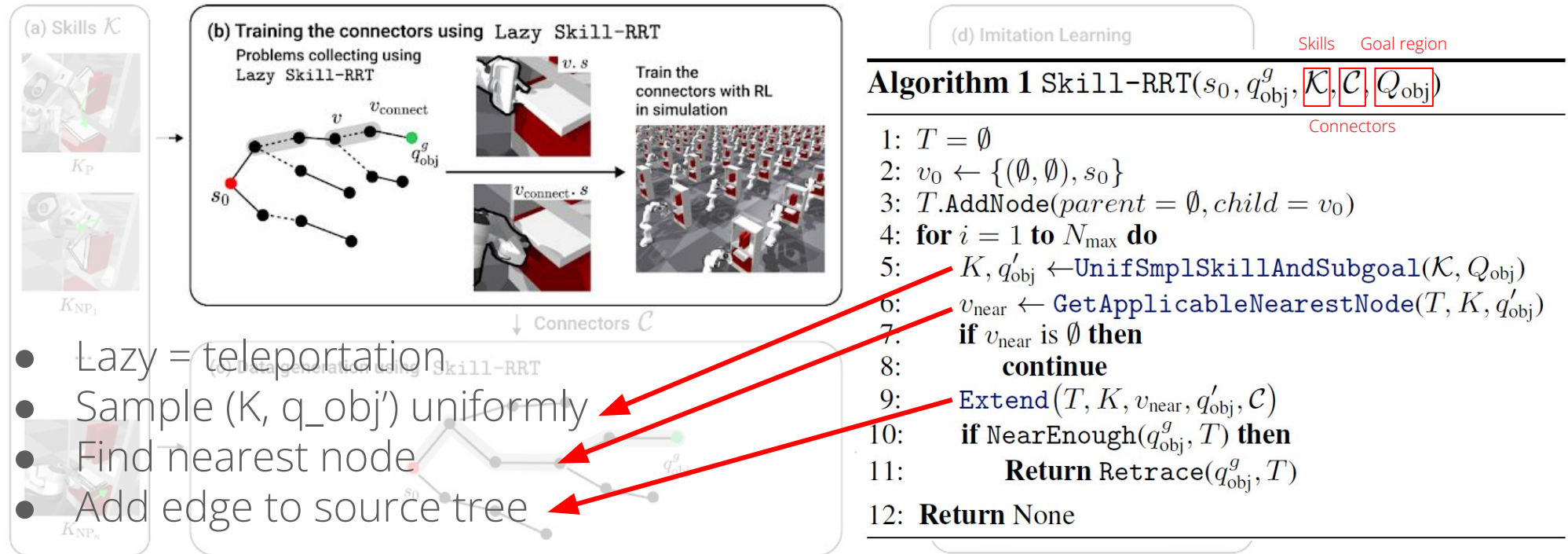


High-level: (1) **Skill cards** => (2) **Skill-RRT** (w/ or w/o connector) => (3) **Distill** into 1 policy

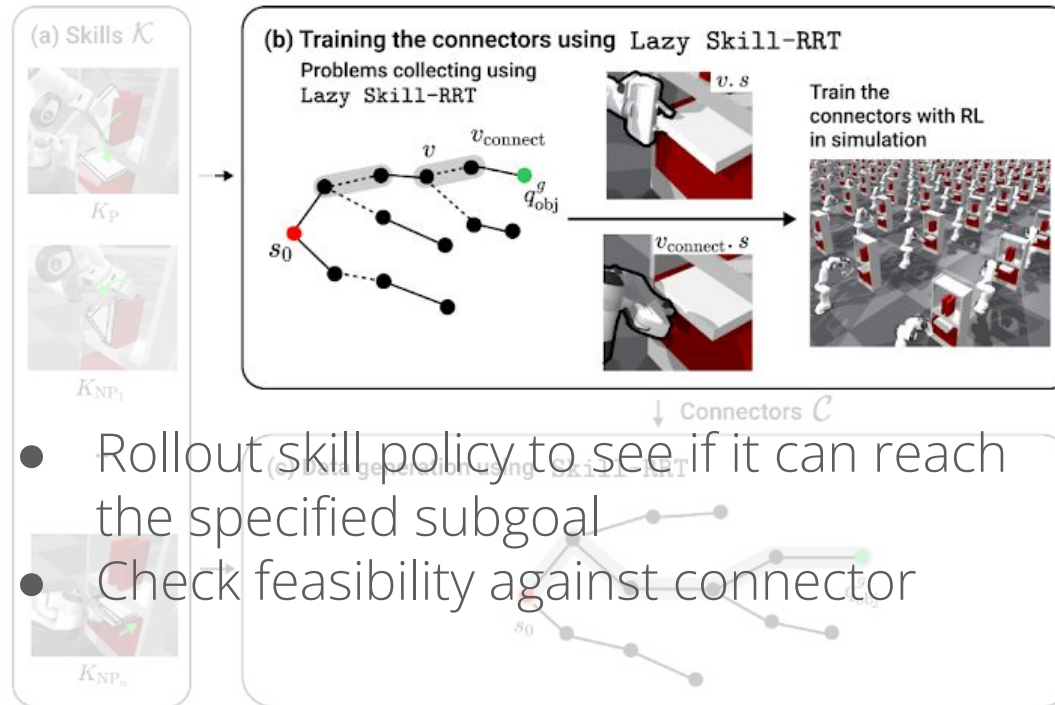
How it works



How it works



How it works

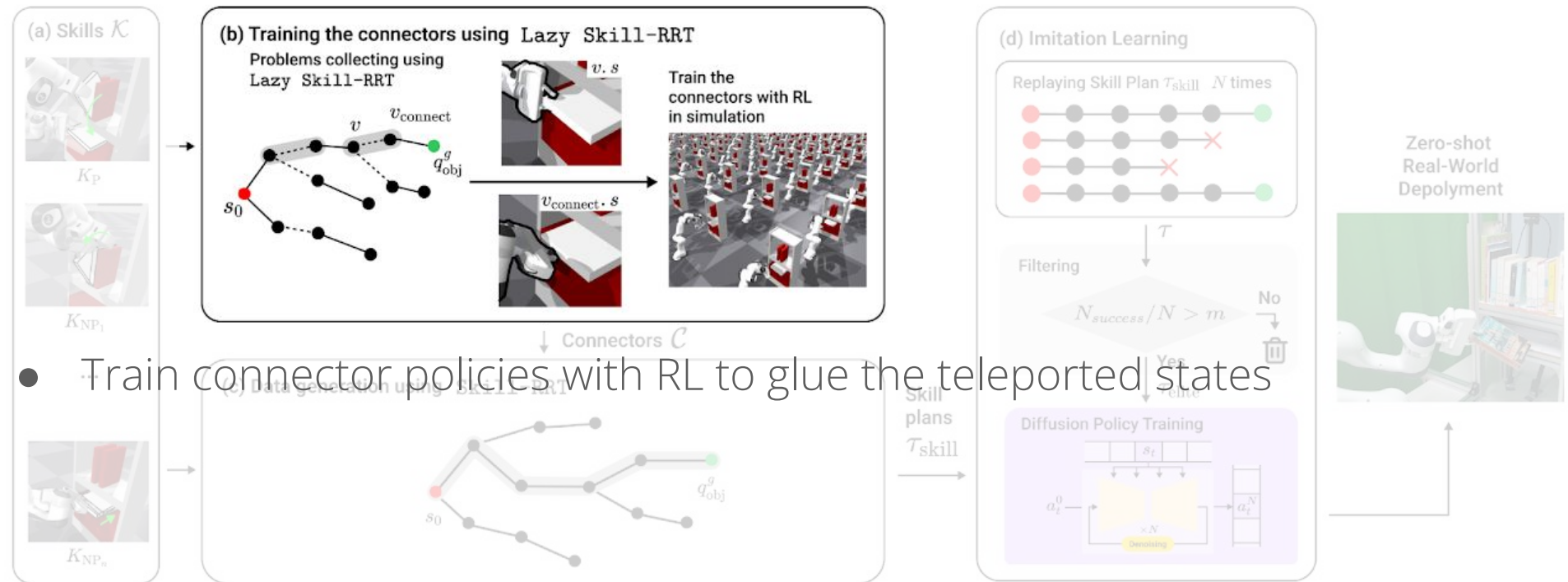


- Rollout skill policy to see if it can reach the specified subgoal
- Check feasibility against connector

Algorithm 2 $\text{Extend}(T, K, v, q'_{\text{obj}}, \mathcal{C})$

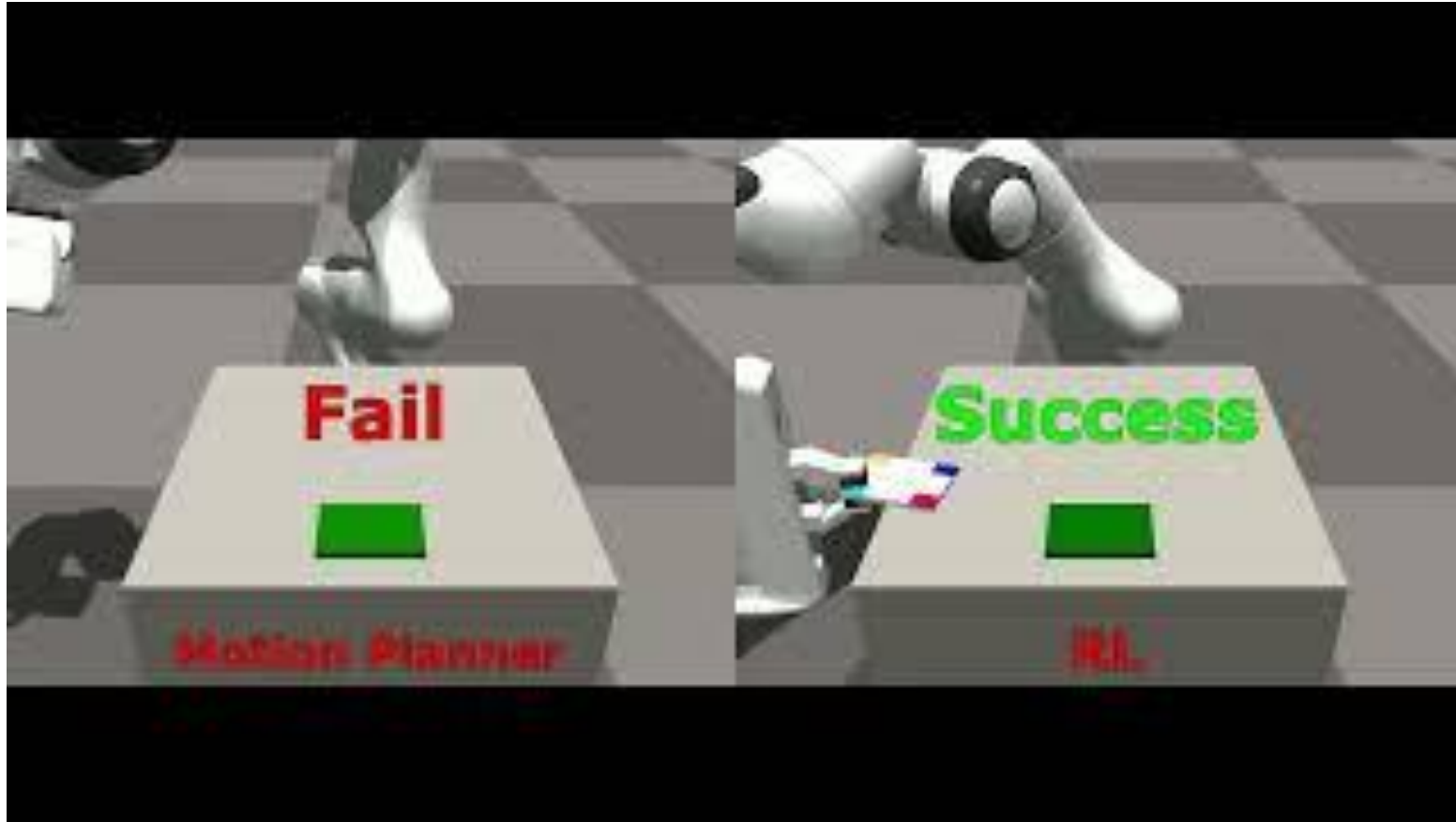
- 1: $q'_r \leftarrow \text{ComputePreSkillConfig}(K, v.s.q_{\text{obj}}, q'_{\text{obj}})$
- 2: $v_{\text{connect}} \leftarrow \text{ComputeConnectingNode}(q'_r, K, \mathcal{C}, v)$
- 3: $s' \leftarrow f_{\text{sim}}(v_{\text{connect}}.s, K.\pi(v_{\text{connect}}.s; q'_{\text{obj}}))$
- 4: **if** Failed(s', q'_{obj}) **then**
- 5: **return**
- 6: $v' \leftarrow (K.\pi_{\text{post}}, q'_{\text{obj}}, s')$
- 7: $T.\text{Add}(\text{parent} = v, \text{child} = v_{\text{connect}})$
- 8: $T.\text{Add}(\text{parent} = v_{\text{connect}}, \text{child} = v')$

How it works



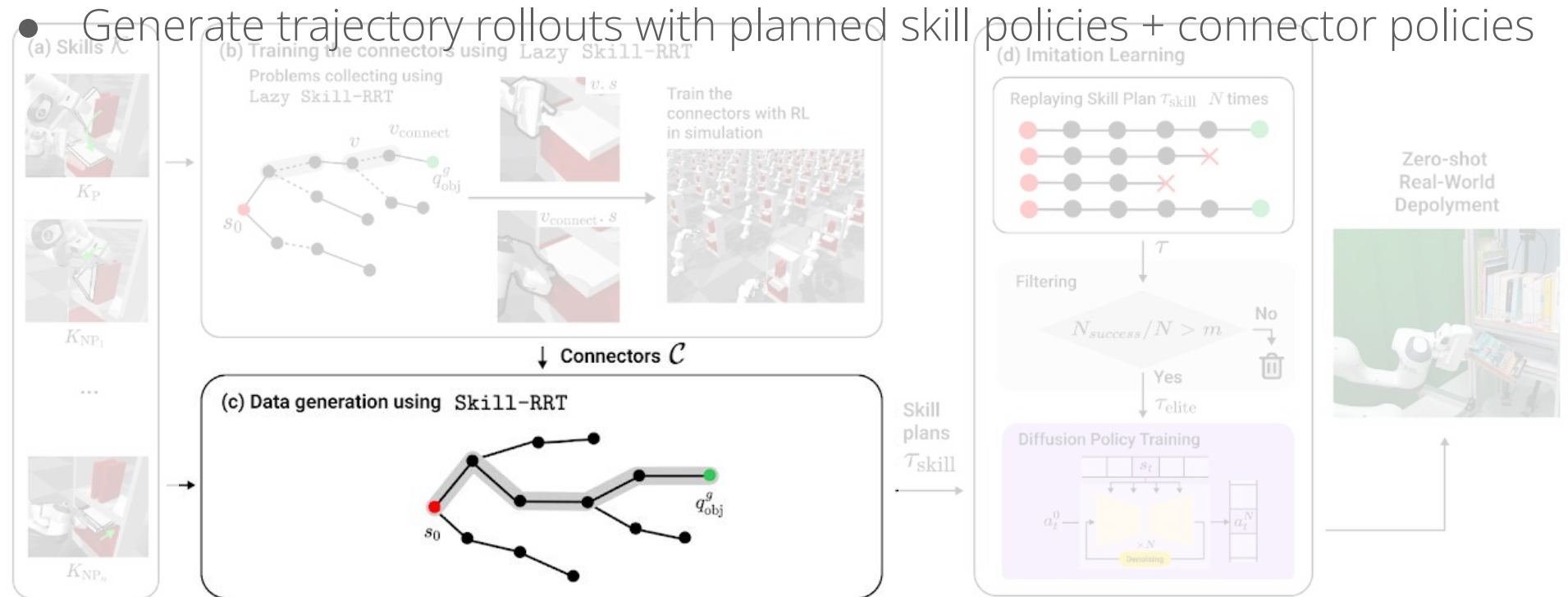
- Train connector policies with RL to glue the teleported states

Why is connector necessary



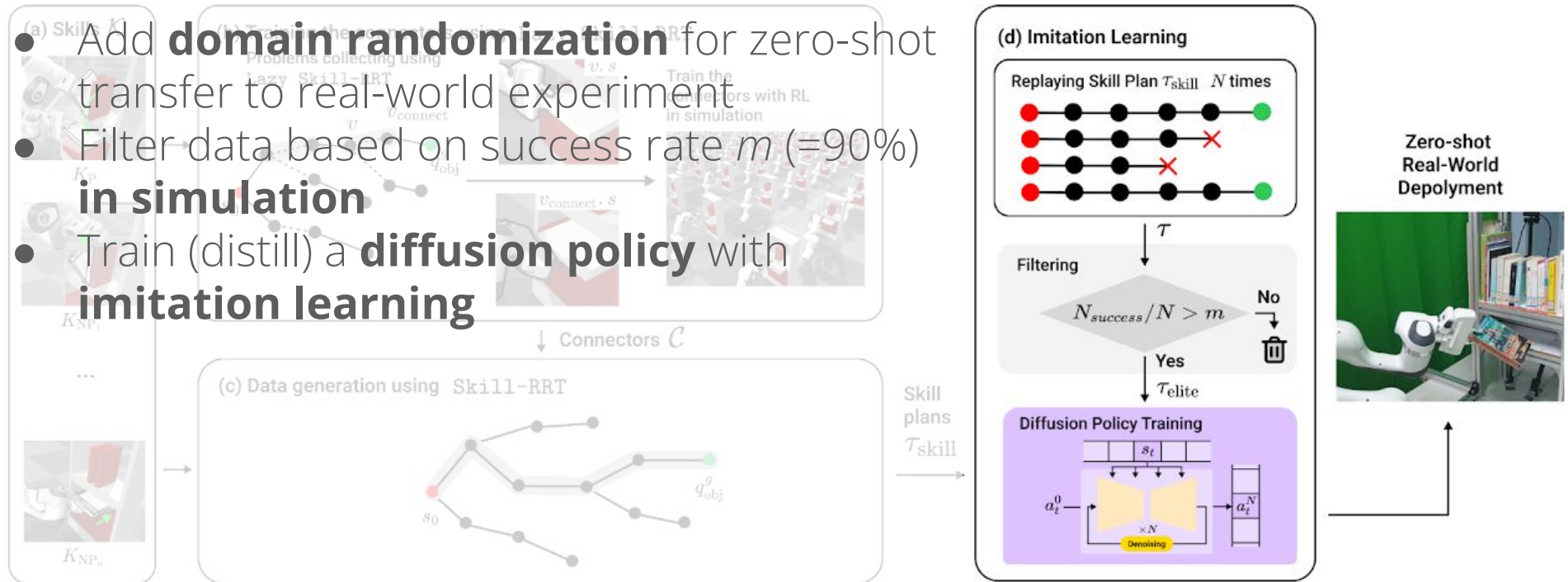
How it works

- Generate trajectory rollouts with planned skill policies + connector policies



How it works

- Add **domain randomization** for zero-shot transfer to real-world experiment
- Filter data based on success rate m ($=90\%$) **in simulation**
- Train (distill) a **diffusion policy** with **imitation learning**



Experiments

		Components			Problem Domain						
		Method	Action Type	Use \mathcal{K}	Card Flip		Bookshelf		Kitchen		
					Success rate (%)	Computation time (s)	Success rate (%)	Computation time (s)	Success rate (%)	Computation time (s)	
		PPO [32]	Flat RL	Low-level action	✗	0.0	N/A	0.0	N/A	0.0	N/A
Hierarchical RL		HLPS [31]	Hierarchical RL	Low-level action	✗	0.0	N/A	0.0	N/A	0.0	N/A
PAMDP		MAPLE [24]	Hierarchical RL	Skill & Parameter	✓	0.0	N/A	78.0	5.3	0.0	N/A
		Skill-RRT	Planning	Skill & Parameter	✓	39.0	85.3 ± 48.7	66.0	79.2 ± 67.1	64.0	121 ± 39.5
		SPIN-w/o-filtering	Planner distilled via IL	Low-level action	✓	82.0	2.68 ± 0.61	83.0	2.93 ± 2.05	87.0	3.02 ± 0.65
		SPIN	Planner distilled via IL	Low-level action	✓	95.0	2.68 ± 0.61	93.0	2.93 ± 2.05	98.0	3.02 ± 0.65

Table 1: Description of baselines (Left) and performance metrics (Right) with three different random seeds for each domain (standard deviations are reported in Table 25).

- 80% success rate in 3 real-world experiments

Limitations

- Yes, it solves a particular long-horizon problem, in a *very clumsy* way
- No, it will **never scale to general robotics** with:
 - Human pre-defined skills and pretrained skill policies
 - prehensile vs non-prehensile: just for a story
 - Pen spin — pre or non-pre?
- Object pose: what about symmetric geometry?
 - Define pose for a perfect sphere
- **Skill card**: what if it's off from the true capabilities of skill policies?
 - Skill-RRT only sees cards, not “looking at policy weights”
 - Gap between what's on the paper and what's actually achievable
- **0 generalizability** to varying object, env — acknowledged by authors
- Skill-RRT is **super time-consuming** — acknowledged by authors
- Connector assumes **static object** — acknowledged by authors
 - Throw & catch impossible

How we can improve it

- Skill card: update skill card as Skill-RRT tests skill policies
- Planning in the latent space
 - Skills should be **discovered/emergent** from data, not pre-defined
 - Skill learning, credit assignment, PAMDP, ...
 - Skills should be applicable to novel tasks/envs/objects/...
 - Planner should organically understand skill token/action chunk/object/env/...
- Thoughts?