Model-based Offline RL for dynamic manipulation

2021.10.29

Dynamics & Control /Kyoungyeon Choi

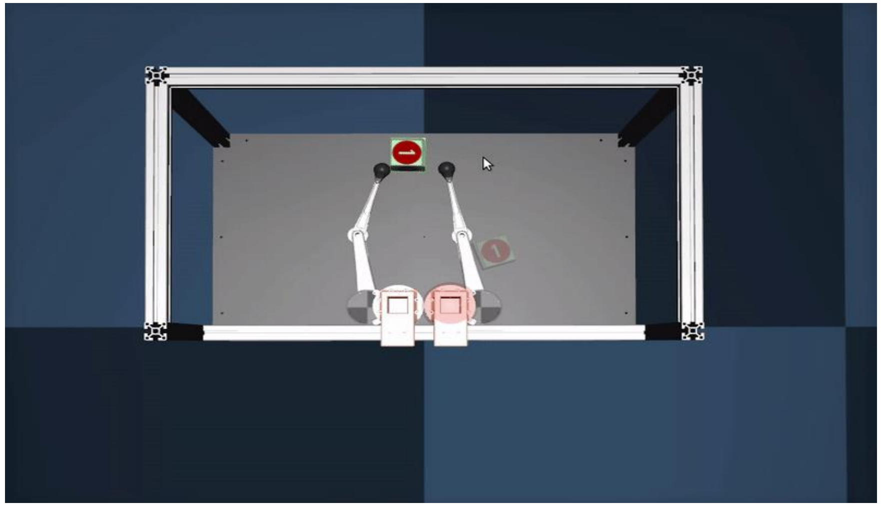
NAVER LABS

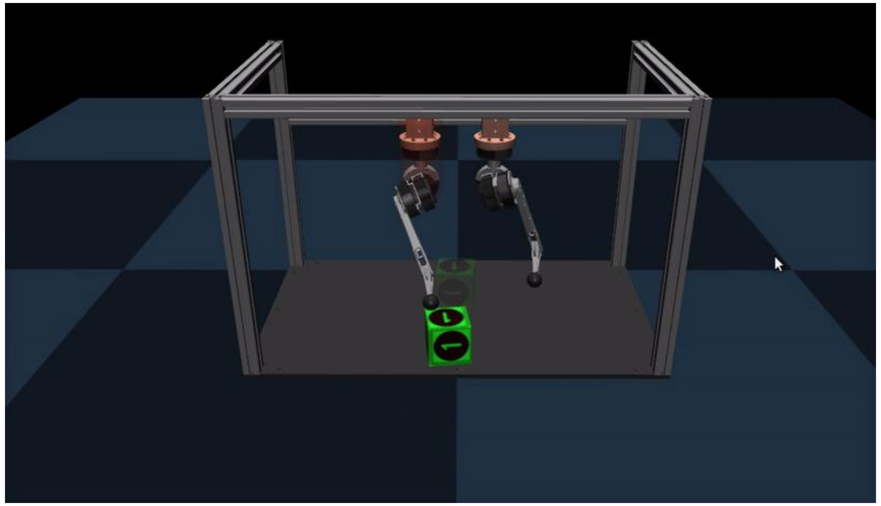
Contents

1. Motivation

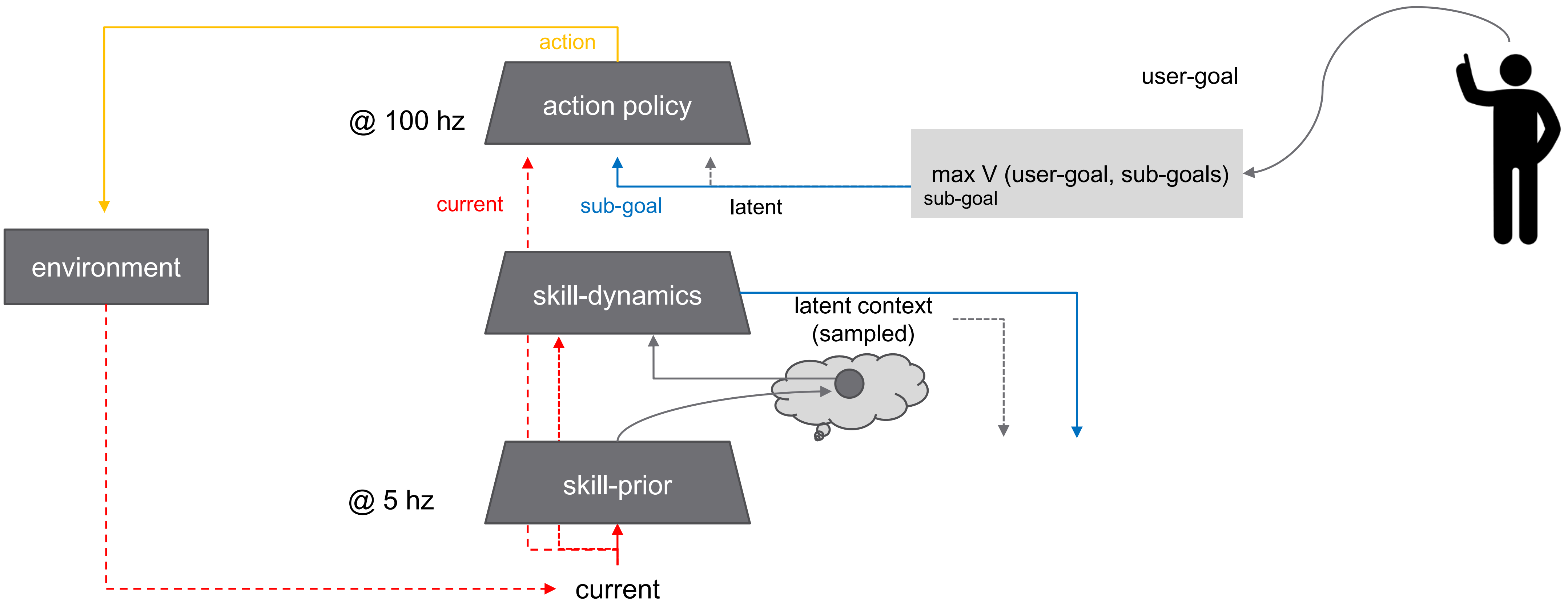
Why offline RL?

Why offline RL?





Why offline RL?

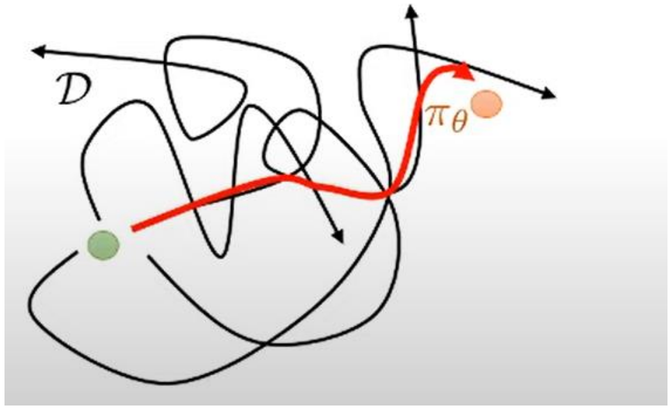


Why offline RL?

Proposed Method | LMP | GCBC

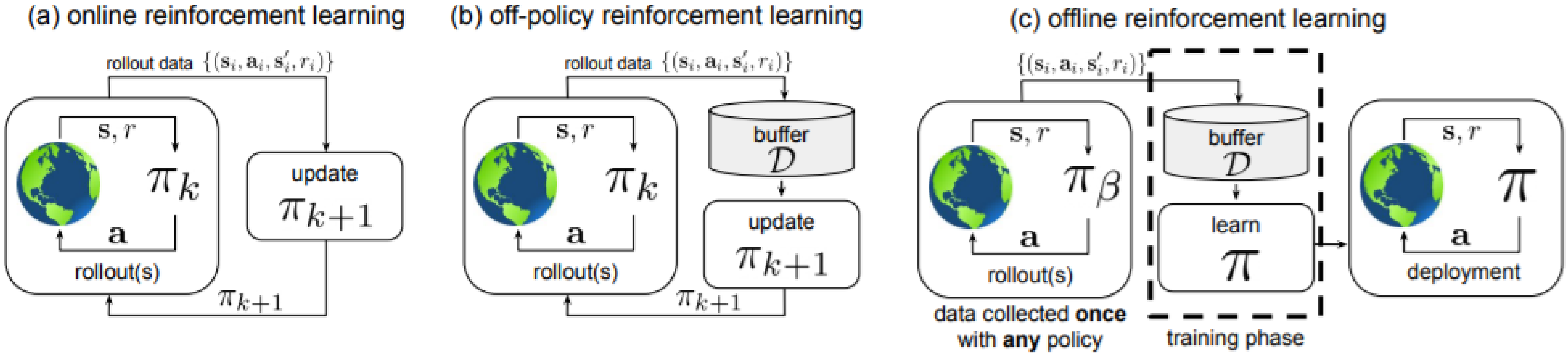
Why offline RL?

- No planning required at test time (offline precomputation) .
- Can be applied to long horizon tasks.
- Can work in sparse reward tasks.
- Do not need additional interaction.
- Data can come from a variety of sources.



Levine, Sergey et al. “Offline Reinforcement Learning: Tutorial, Review, and Perspectives on Open Problems.” *ArXiv* abs/2005.01643 (2020)

Why offline RL?



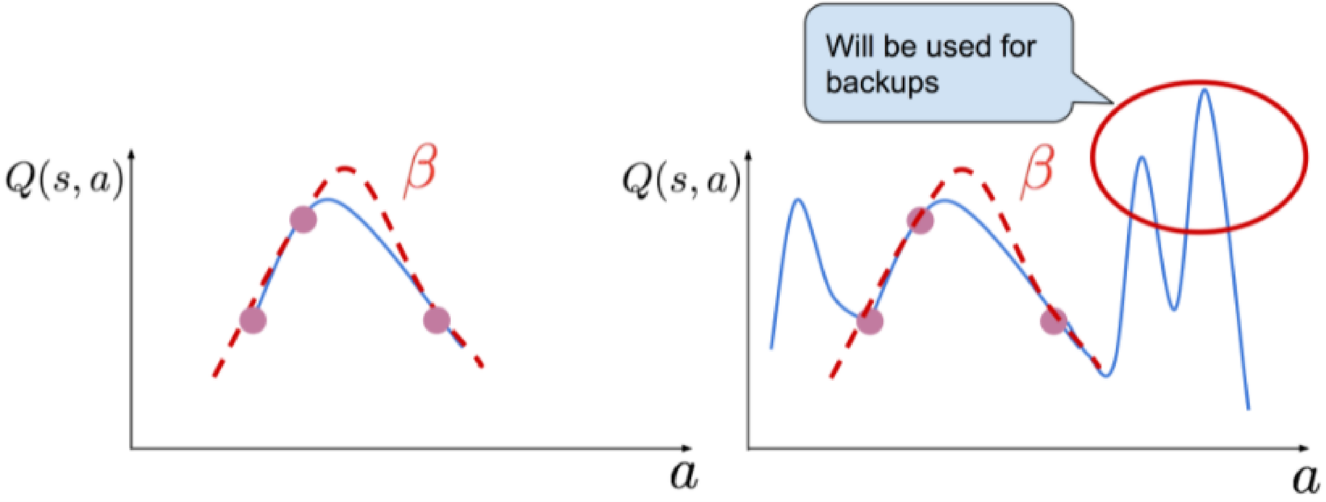
Levine, Sergey et al. "Offline Reinforcement Learning: Tutorial, Review, and Perspectives on Open Problems." *ArXiv* abs/2005.01643 (2020)

1. Motivation

Difficulties in offline RL

Difficulties in offline RL

- No possibility of improving exploration.
- Distributional shift.



$$Q(\mathbf{s},\mathbf{a}) \leftarrow r(\mathbf{s},\mathbf{a}) + E_{\mathbf{a}'\sim\pi_{\text{new}}}[Q(\mathbf{s}',\mathbf{a}')]$$

$$\pi_{\text{new}}(\mathbf{a}|\mathbf{s}) = \arg\max_{\pi} E_{\mathbf{a}\sim\pi(\mathbf{a}|\mathbf{s})}[Q(\mathbf{s},\mathbf{a})]$$

$$\text{expect good accuracy when } \pi_\beta(\mathbf{a}|\mathbf{s}) = \pi_{\text{new}}(\mathbf{a}|\mathbf{s})$$

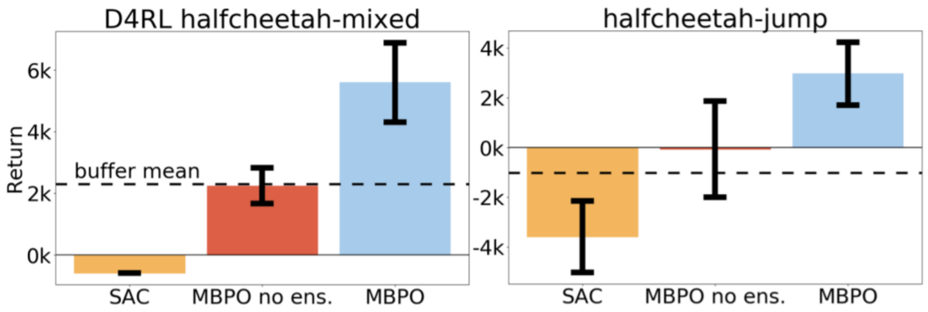
2. Approach & Contributions

Related works

$$\hat{Q}^{k+1} \leftarrow \arg\min_{Q} \alpha \cdot \left(\mathbb{E}_{\mathbf{s}\sim\mathcal{D},\mathbf{a}\sim\mu(\mathbf{a}|\mathbf{s})}\left[Q(\mathbf{s},\mathbf{a})\right] - \mathbb{E}_{\mathbf{s}\sim\mathcal{D},\mathbf{a}\sim\hat{\pi}_{\beta}(\mathbf{a}|\mathbf{s})}\left[Q(\mathbf{s},\mathbf{a})\right]\right) + \frac{1}{2}\mathbb{E}_{\mathbf{s},\mathbf{a},\mathbf{s}'\sim\mathcal{D}}\left[\left(Q(\mathbf{s},\mathbf{a}) - \hat{\mathcal{B}}^{\pi}\hat{Q}^{k}(\mathbf{s},\mathbf{a})\right)^2\right].$$

$$\min_{Q}\max_{\mu} \alpha \left(\mathbb{E}_{\mathbf{s}\sim\mathcal{D},\mathbf{a}\sim\mu(\mathbf{a}|\mathbf{s})}\left[Q(\mathbf{s},\mathbf{a})\right] - \mathbb{E}_{\mathbf{s}\sim\mathcal{D},\mathbf{a}\sim\hat{\pi}_{\beta}(\mathbf{a}|\mathbf{s})}\left[Q(\mathbf{s},\mathbf{a})\right]\right) + \frac{1}{2}\mathbb{E}_{\mathbf{s},\mathbf{a},\mathbf{s}'\sim\mathcal{D}}\left[\left(Q(\mathbf{s},\mathbf{a}) - \hat{\mathcal{B}}^{\pi_k}\hat{Q}^{k}(\mathbf{s},\mathbf{a})\right)^2\right] + \mathcal{R}(\mu) \quad (\text{CQL}(\mathcal{R})).$$

Kumar, Aviral et al. "Conservative Q-Learning for Offline Reinforcement Learning." *ArXiv* abs/2006.04779 (2020)



Algorithm 1 Framework for Model-based Offline Policy Optimization (MOPO) with Reward Penalty

Require: Dynamics model $\widehat{T}$ with admissible error estimator $u(s,a)$; constant λ.

1: Define $\tilde{r}(s,a) = r(s,a) - \lambda u(s,a)$. Let $\widetilde{M}$ be the MDP with dynamics $\widehat{T}$ and reward $\tilde{r}$.

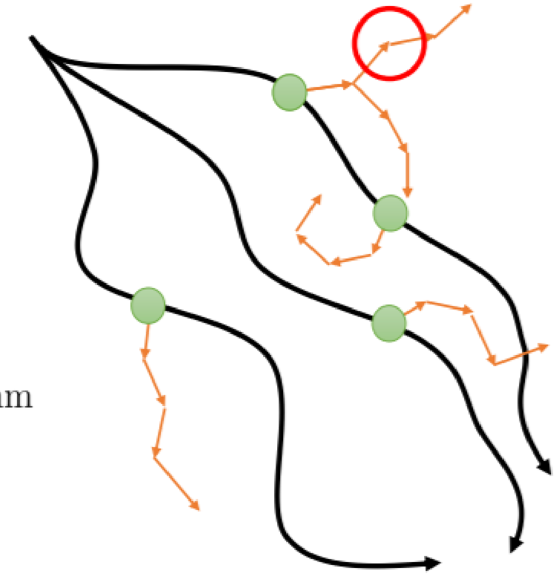
2: Run any RL algorithm on $\widetilde{M}$ until convergence to obtain $\hat{\pi} = \text{argmax}_{\pi}\eta_{\widetilde{M}}(\pi)$

solution: "punish" the policy for exploiting

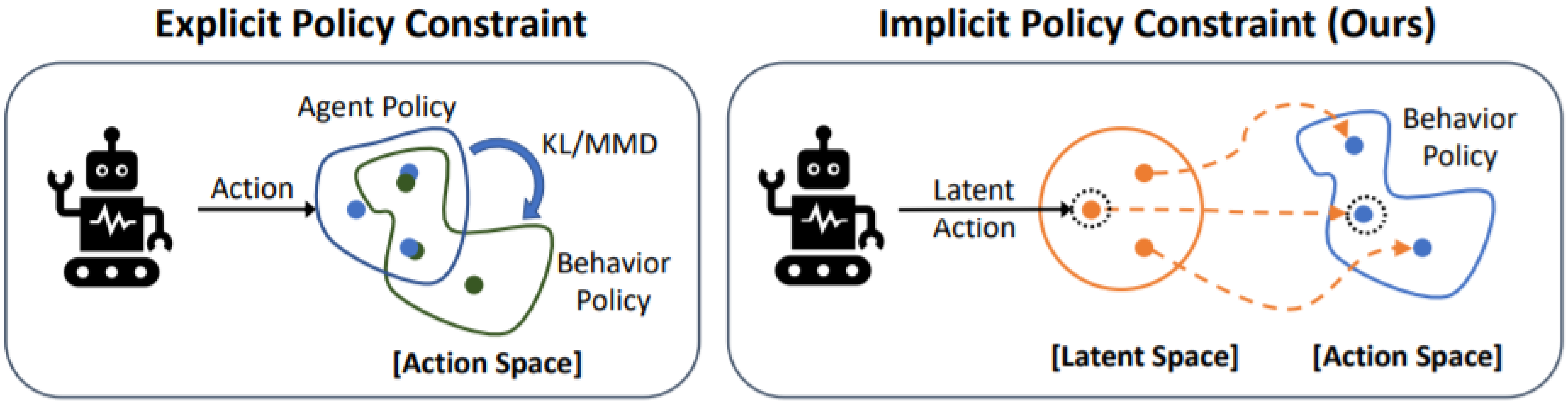
$$\tilde{r}(\mathbf{s},\mathbf{a}) = r(\mathbf{s},\mathbf{a}) - \lambda u(\mathbf{s},\mathbf{a})$$

uncertainty penalty

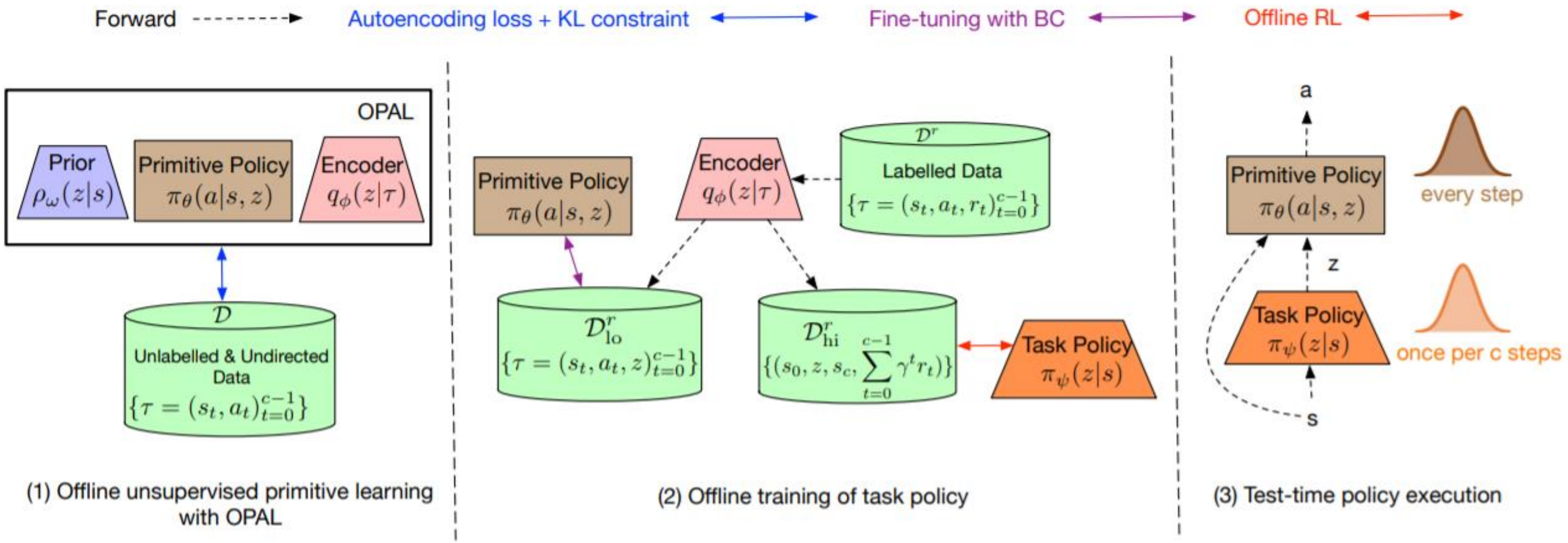
...and then use any existing model-based RL algorithm



Yu, Tianhe et al. "MOPO: Model-based Offline Policy Optimization." *ArXiv* abs/2005.13239 (2020)

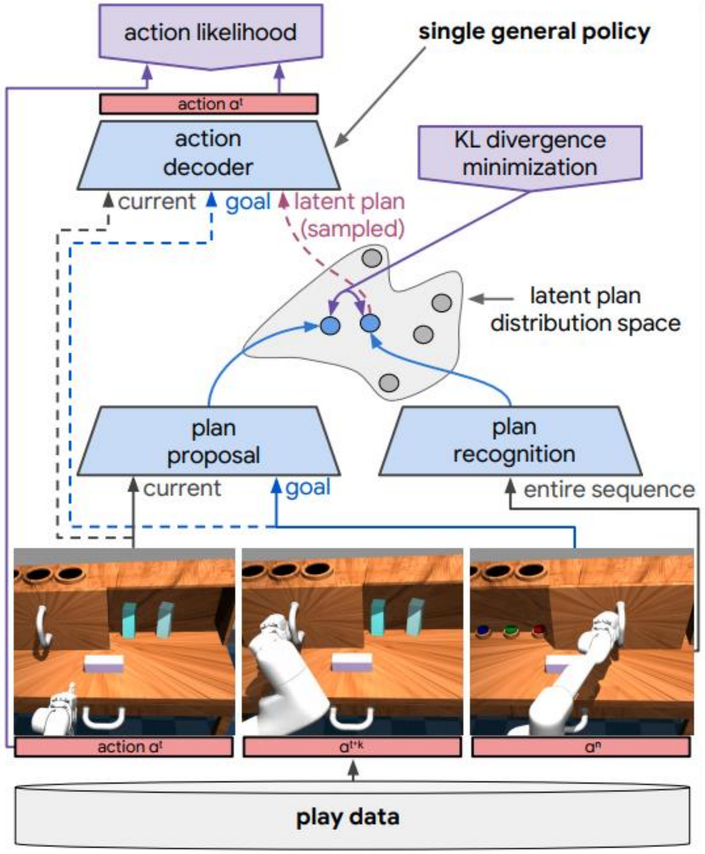


Zhou, Wenxuan et al. “PLAS: Latent Action Space for Offline Reinforcement Learning.” *CoRL* (2020).

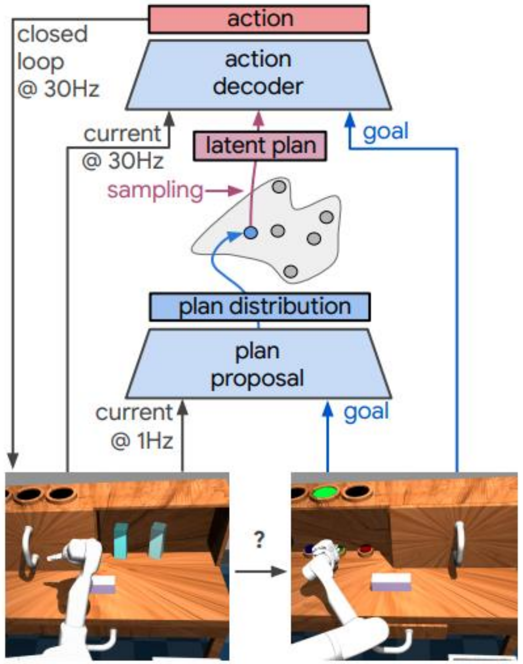


Ajay, Anurag et al. "OPAL: Offline Primitive Discovery for Accelerating Offline Reinforcement Learning." *ArXiv* abs/2010.13611 (2021)

Related works

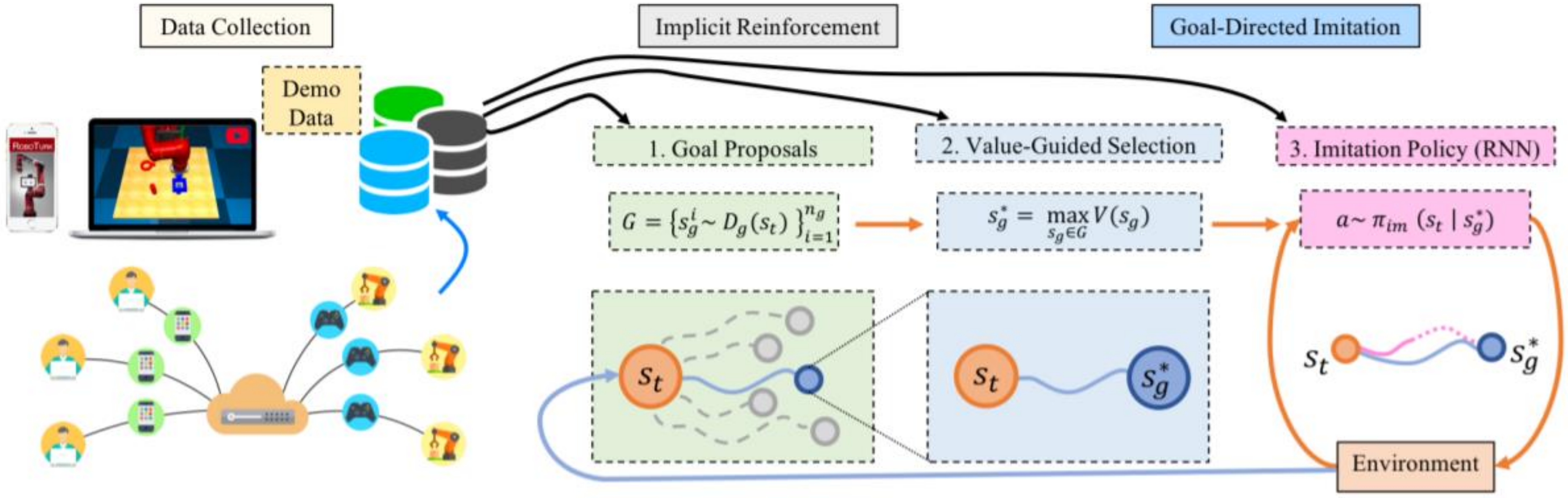


(a) **Play-LMP** training.



(b) **Task-agnostic policy inference**

Lynch, Corey, et al. "Learning latent plans from play." *Conference on Robot Learning*. PMLR, 2020.

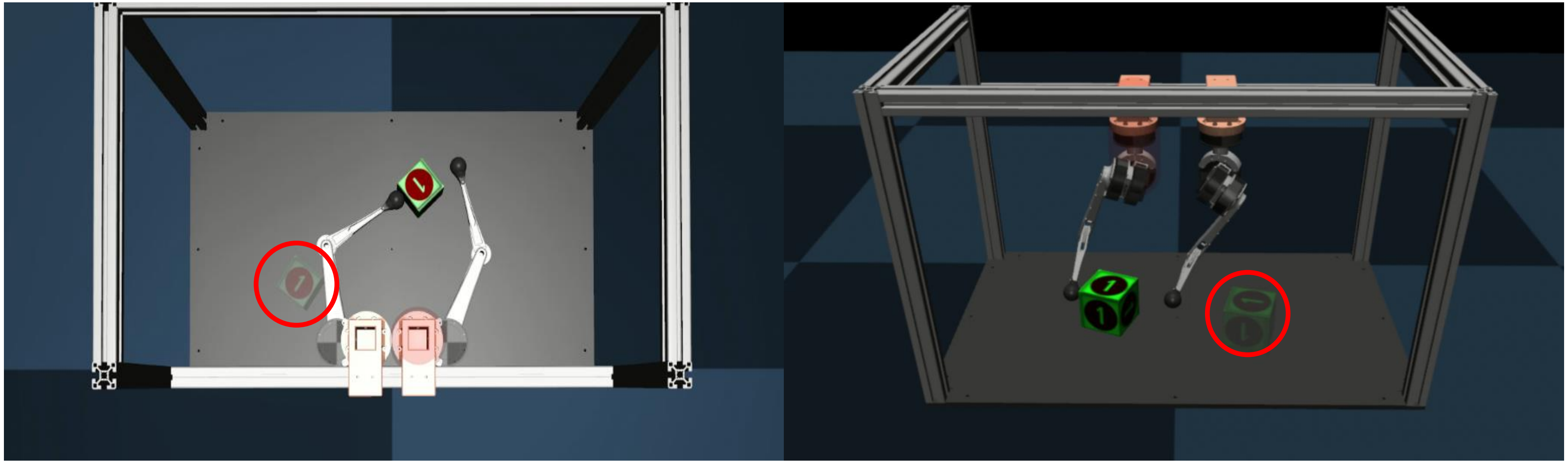


Mandlekar, Ajay et al. "IRIS: Implicit Reinforcement without Interaction at Scale for Learning Control from Offline Robot Manipulation Data." *2020 IEEE International Conference on Robotics and Automation (ICRA)* (2020): 4414-4420.

2. Approach & Contributions

Approach

Approach



1. 2D manipulation task

State $S \in R^{29}$

$$S = (p_{L_{ee}}, cos\theta_L, sin\theta_L, \omega_{Lee}, v_{Lee},$$
$$p_{R_{ee}}, cos\theta_R, sin\theta_R, \omega_{Ree}, v_{Ree},$$
$$p_{box}, cos\theta_{box}, sin\theta_{box}, \omega_{box}, v_{box}, p_{Lrel}, p_{Rrel})$$

Goal $G \in R^4$

$$G = (p_{box}, cos\theta_{box}, sin\theta_{box})$$

Action $A \in R^4$

$$A = \Delta q$$

2. 3D manipulation task

State $S \in R^{78}$

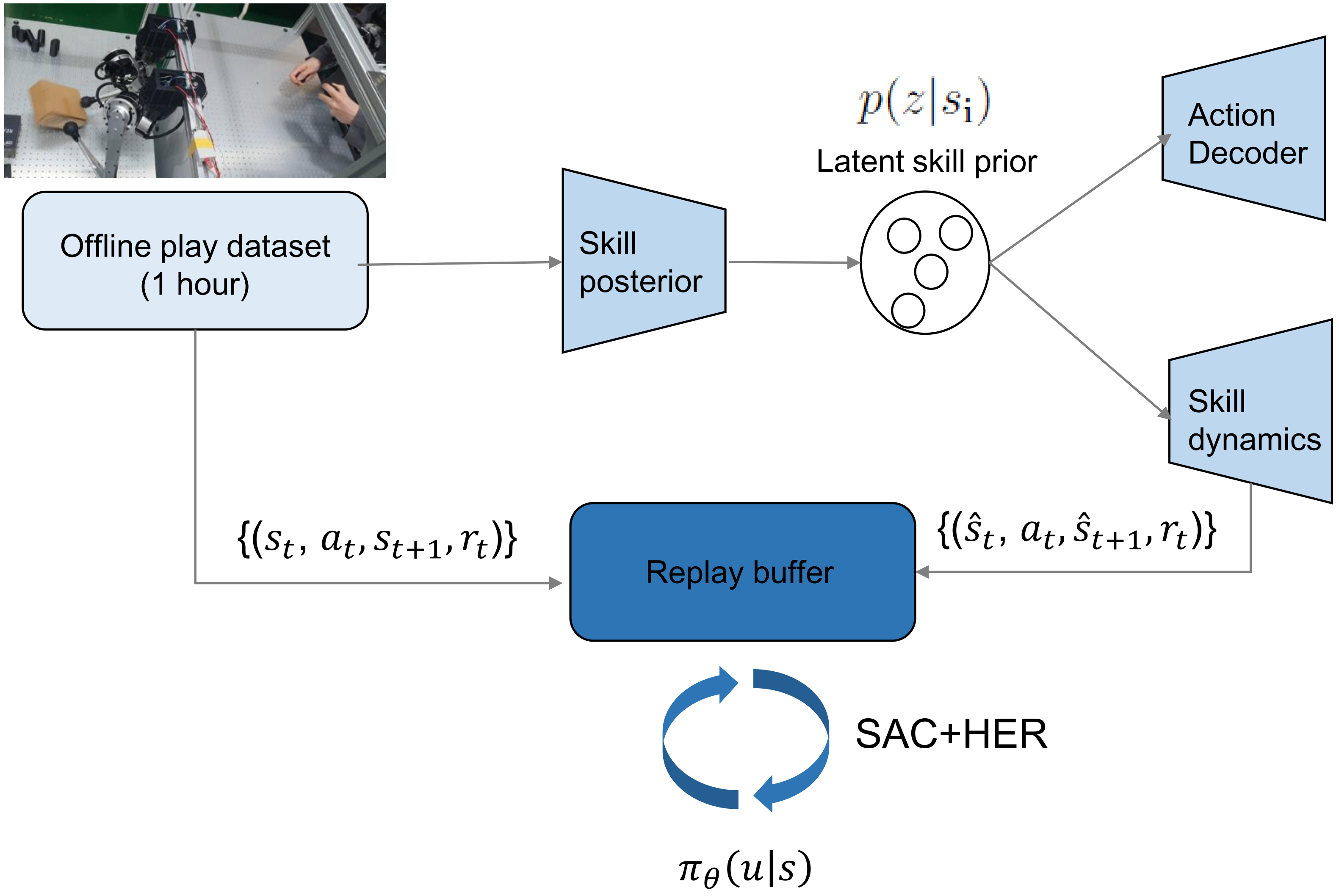
$$S = (q_{Lee}, \dot{q}_{Lee}, q_{Ree}, \dot{q}_{Ree}, q_{box}, \dot{q}_{box}, q_{Lrel}, q_{Rrel})$$

Goal $G \in R^{12}$

$$G = (q_{box}) = (rotation\ matrix, xyz\ position)$$

Action $A \in R^8$

$$A = \Delta q$$


Offline play dataset
(1 hour)
Skill
posterior
$p(z|s_i)$
Latent skill prior
Action
Decoder
Skill
dynamics
$\{(s_t, a_t, s_{t+1}, r_t)\}$
Replay buffer
$\{(\hat{s}_t, a_t, \hat{s}_{t+1}, r_t)\}$
SAC+HER
$\pi_\theta(u|s)$

MDP for 3D Goal-conditioned policy

State $S \in R^{80}$

$$S = (s, G)$$

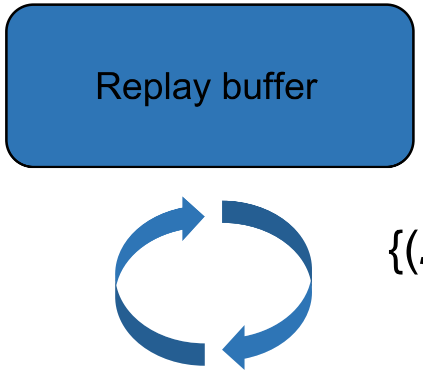
Action $u \in R^{16}, u_{max} = 1$

$$P(z|S) = N(\mu,\ \sigma)$$

$$z = \mu + \sigma * u$$

Binary Reward $R \in R^{1}$

$$\begin{cases} R = -1 \ if\ not\ done \\ R = 0 \qquad if\ done \end{cases}$$



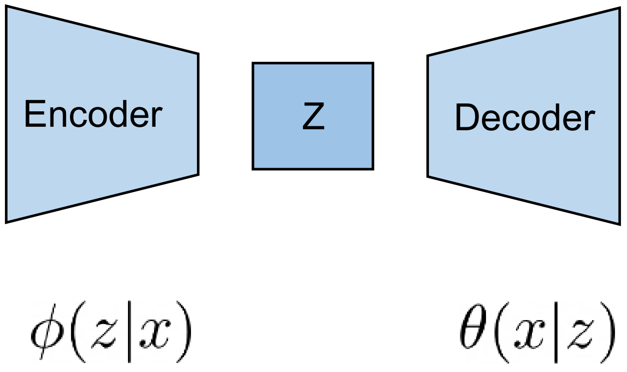
$$\{(S, u, S', R)\}$$

$$\pi_\theta(u|s)$$

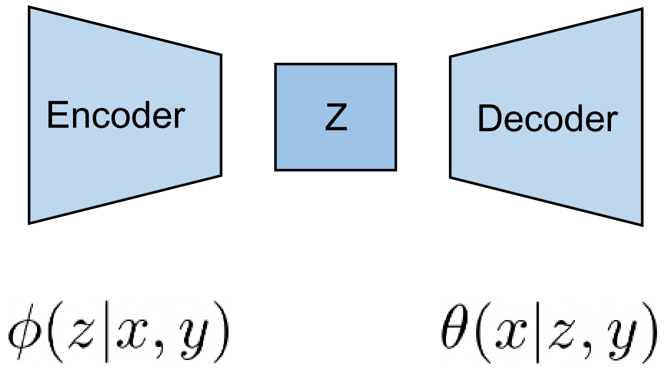
3. Method

CVAE skill learning

CVAE skill learning



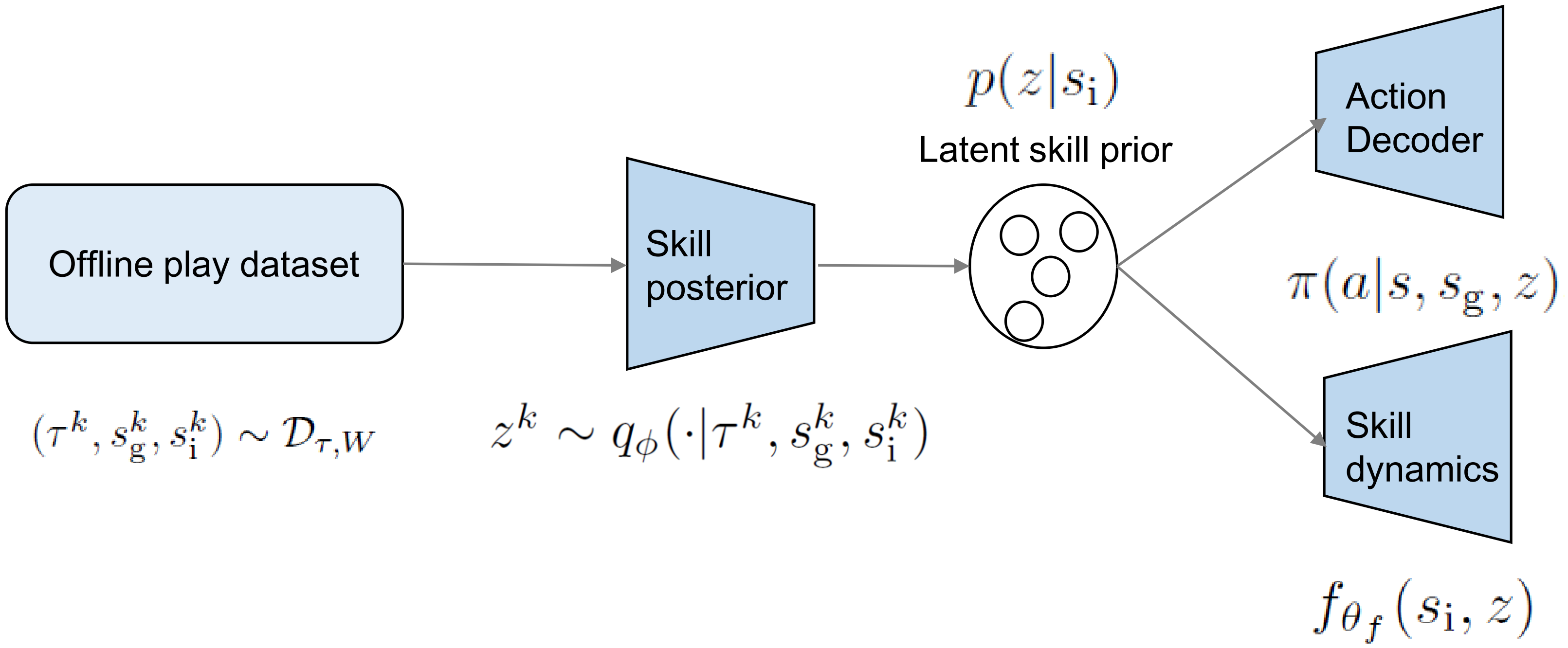
VAE



CVAE

Sohn, Kihyuk, Honglak Lee, and Xinchen Yan. "Learning structured output representation using deep conditional generative models." *Advances in neural information processing systems* 28 (2015): 3483-3491.

CVAE skill learning



$$\mathcal{L}_{\mathrm{KL}} = D_{\mathrm{KL}}(q_\phi(\cdot|\tau, s_{\mathrm{g}}, s_{\mathrm{i}}) \| p_\psi(\cdot|s_{\mathrm{i}})),$$

$$\mathcal{L}_{\mathrm{rec}} \triangleq \sum_{(s,a)\in\tau} \|a - \pi_{\theta_\pi}(s, s_{\mathrm{g}}, z)\|^2 + \|s_{\mathrm{g}} - f_{\theta_f}(s_{\mathrm{i}}, z)\|^2$$

$$\mathcal{J}_{\mathrm{cVAE}} = \mathbb{E}_{\mathcal{D}_{\tau,W}}\left[\mathbb{E}_{q_\phi(z|\tau,s_{\mathrm{g}},s_{\mathrm{i}})}\left[\mathcal{L}_{\mathrm{rec}}\right] + \beta\mathcal{L}_{KL}\right]$$

CVAE skill learning

$$p(\tau, s_g|s_i) = \int_{\mathcal{Z}} p(\tau|s_g, s_i, z)\ p(s_g|s_i, z)\ p(z|s_i)\ dz$$
$$= \int_{\mathcal{Z}} \prod_{(s,a,s')\in\tau} \Big\{ \underbrace{\pi(a|s, s_g, z)}_{\text{skill-policy}}\ p(s'|s,a) \Big\} \underbrace{p(s_g|s_i, z)}_{\text{skill-dynamics}}\ \underbrace{p(z|s_i)}_{\text{skill-prior}}\ dz,$$

$$\min_{\theta_\pi, \theta_f, \phi, \psi} \mathbb{E}_{\mathcal{D}_{\tau,W}} \left[\mathbb{E}_{q_\phi(z|\tau, s_g, s_i)} \left[\mathcal{L}_{\text{rec}} \right] \right]$$
$$\text{s.t. } \hat{p}_{\mathcal{D},\phi}(\cdot|s_i) = p_\psi(\cdot|s_i), \text{ for all } s_i \in \mathcal{D}_{\tau,W}$$

$$\hat{p}_{\mathcal{D},\phi}(z|s_i) \triangleq \mathbb{E}_{\tau, s_g \sim \mathcal{D}_{\tau,W}} \left[q_\phi(z|\tau, s_g, s_i) \right]$$

$$\mathcal{L}_{\text{rec}} \triangleq \sum_{(s,a)\in\tau} \|a - \pi_{\theta_\pi}(s, s_g, z)\|^2 + \|s_g - f_{\theta_f}(s_i, z)\|^2$$

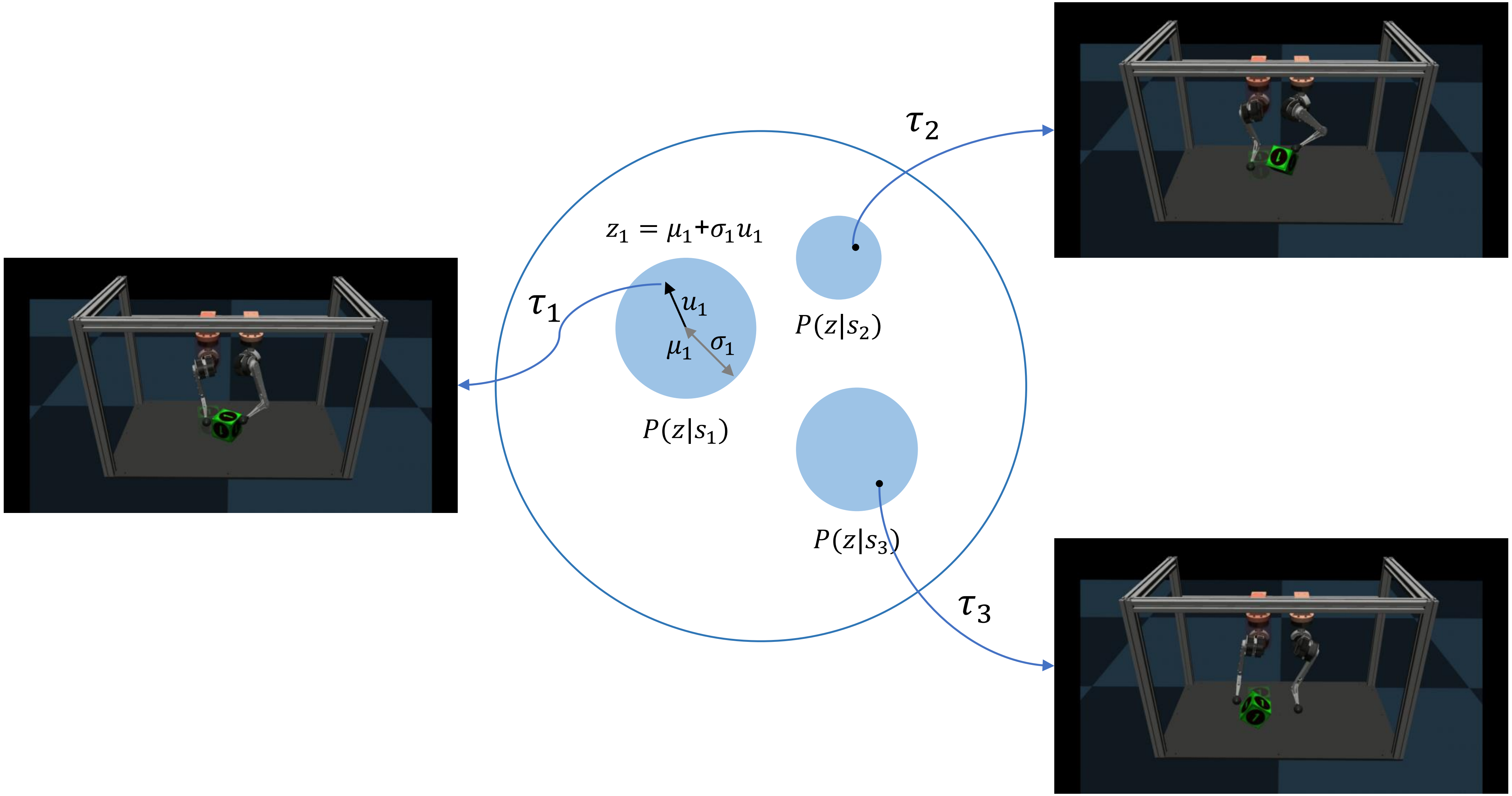
$$\mathcal{L}_{\text{KL}} = D_{\text{KL}}(q_\phi(\cdot|\tau, s_g, s_i) \| p_\psi(\cdot|s_i)),$$

$$\mathcal{J}_{\text{cVAE}} = \mathbb{E}_{\mathcal{D}_{\tau,W}} \left[\mathbb{E}_{q_\phi(z|\tau, s_g, s_i)} \left[\mathcal{L}_{\text{rec}} \right] + \beta \mathcal{L}_{KL} \right]$$

Algorithm 1: First Stage Skill Learning

Input : Offline trajectory dataset $\mathcal{D}_{\tau,W}$
Initialize the parameters, $\theta_\pi, \theta_f, \phi, \psi_{\text{cvae}}$
while *not converged* **do**
  Sample $(\tau^k, s_g^k, s_i^k) \sim \mathcal{D}_{\tau,W}$, for $k = 1, \cdots, n_{\text{batch}}$
  Sample $z^k \sim q_\phi(\cdot|\tau^k, s_g^k, s_i^k)$, for $k = 1, \cdots, n_{\text{batch}}$
  Update the model parameters by descending the batched cVAE loss $\mathcal{J}_{\text{cVAE}}$ (9)
end
Output: $\theta_\pi, \theta_f, \phi, (\psi_{\text{cvae}})$

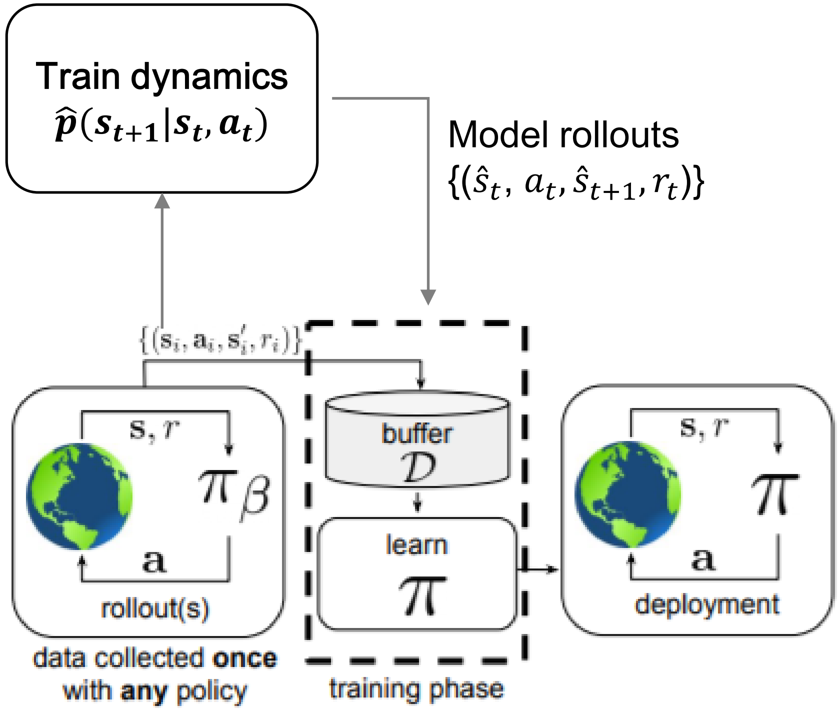
CVAE skill learning



3. Method

Model-based offline RL

Model-based offline RL



$$J_V(\psi) = \mathbb{E}_{\mathbf{s}_t \sim \mathcal{D}} \left[\frac{1}{2} \left(V_\psi(\mathbf{s}_t) - \mathbb{E}_{\mathbf{a}_t \sim \pi_\phi} \left[Q_\theta(\mathbf{s}_t, \mathbf{a}_t) - \log \pi_\phi(\mathbf{a}_t|\mathbf{s}_t) \right] \right)^2 \right]$$

$$J_Q(\theta) = \mathbb{E}_{(\mathbf{s}_t, \mathbf{a}_t) \sim \mathcal{D}} \left[\frac{1}{2} \left(Q_\theta(\mathbf{s}_t, \mathbf{a}_t) - \hat{Q}(\mathbf{s}_t, \mathbf{a}_t) \right)^2 \right]$$

$$J_\pi(\phi) = \mathbb{E}_{\mathbf{s}_t \sim \mathcal{D}} \left[\mathrm{D}_{\mathrm{KL}} \left(\pi_\phi(\cdot | \mathbf{s}_t) \middle\| \frac{\exp(Q_\theta(\mathbf{s}_t, \cdot))}{Z_\theta(\mathbf{s}_t)} \right) \right]$$

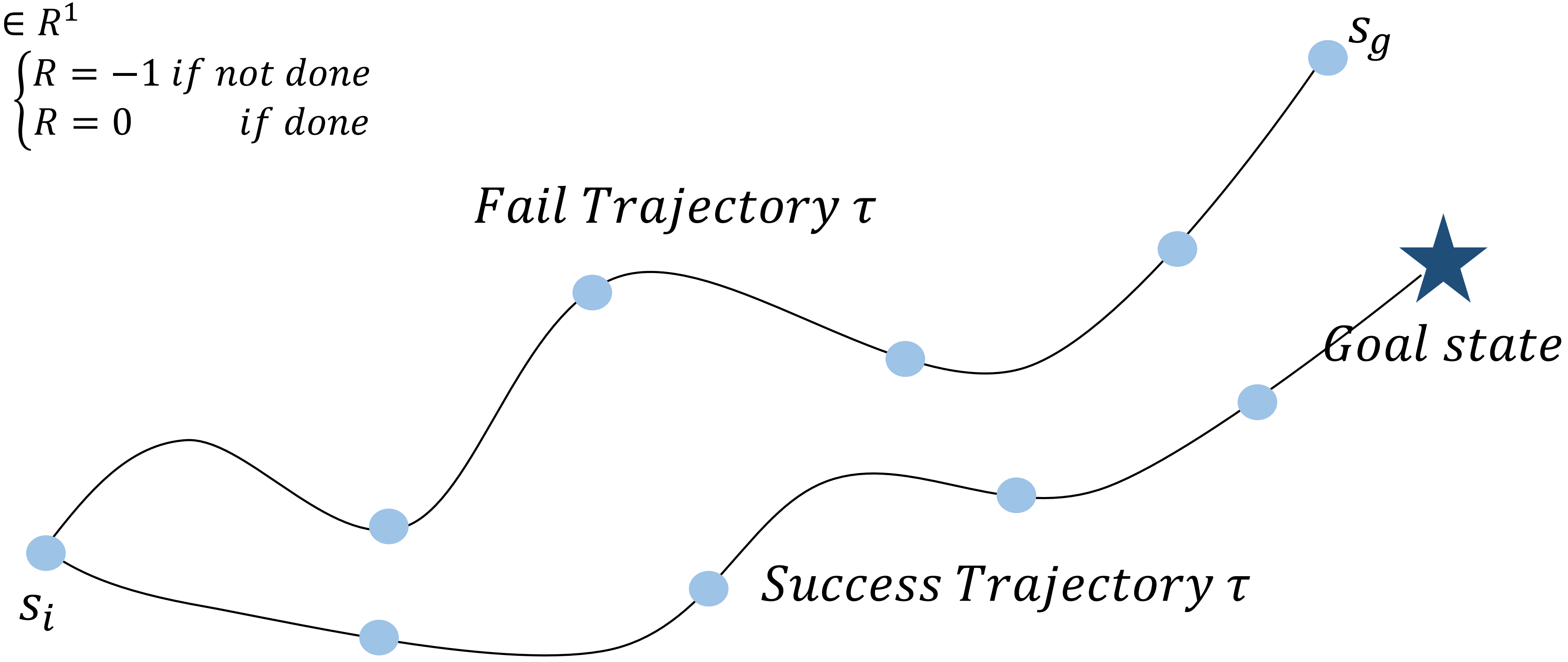
Algorithm 1 Soft Actor-Critic

```
Initialize parameter vectors ψ, ψ̄, θ, φ.
for each iteration do
    for each environment step do
        a_t ~ π_φ(a_t|s_t)
        s_{t+1} ~ p(s_{t+1}|s_t, a_t)
        D ← D ∪ {(s_t, a_t, r(s_t, a_t), s_{t+1})}
    end for
    for each gradient step do
        ψ ← ψ − λ_V ∇̂_ψ J_V(ψ)
        θ_i ← θ_i − λ_Q ∇̂_{θ_i} J_Q(θ_i) for i ∈ {1, 2}
        φ ← φ − λ_π ∇̂_φ J_π(φ)
        ψ̄ ← τψ + (1 − τ)ψ̄
    end for
end for
```

Haarnoja, Tuomas et al. "Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor." *ICML* (2018).

Sparse Reward $R \in R^1$

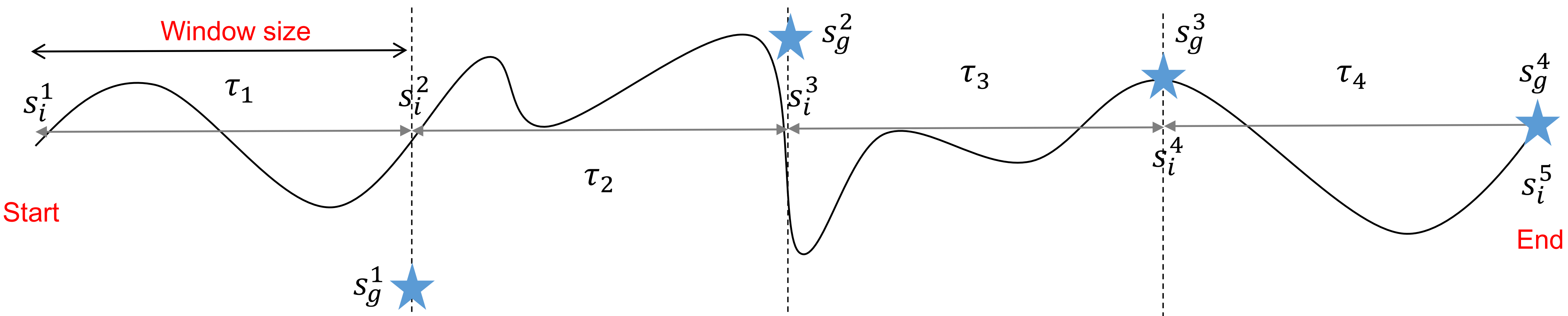
$$\begin{cases} R = -1 \ \textit{if not done} \\ R = 0 \qquad \textit{if done} \end{cases}$$



Andrychowicz, Marcin et al. "Hindsight Experience Replay." *NIPS* (2017).

Model-based offline RL

Offline play dataset



$((s_i^1, s_g^1), u_1, (s_i^2, s_g^1), R_1)$ $((s_i^2, s_g^2), u_2, (s_i^3, s_g^2), R_2)$ $((s_i^3, s_g^3), u_3, (s_i^4, s_g^3),), R_3)$ $((s_i^4, s_g^4), u_4, (s_i^5, s_g^4), R_4)$

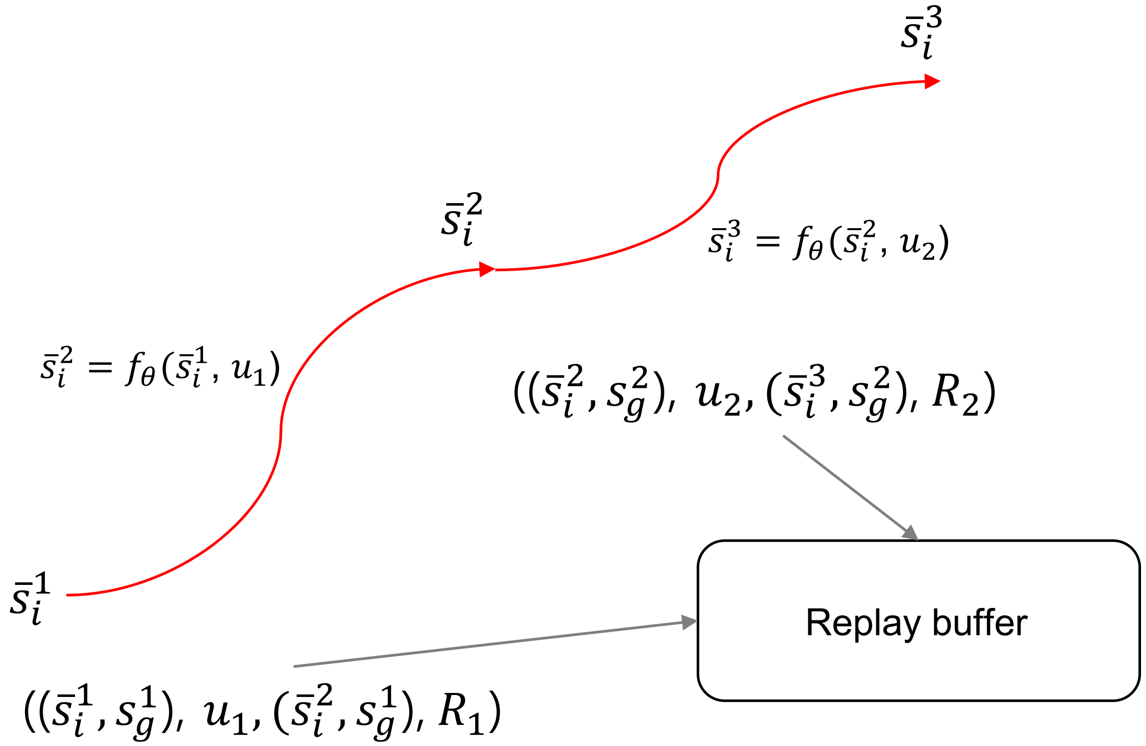
Goal randomization

Goal relabeling(HER)

Replay buffer

Model-based offline RL

Model-based rollouts (if N = 2)



Algorithm 4: Offline Skill Planning: Offline RL

Input : Offline dataset $\mathcal{D}_{\tau,W}$, skill-policy, dynamics, prior, skill posterior: $\pi_{\theta_\pi}, f_{\theta_f}, h_\psi, q_\phi$, hyperparameters $\gamma, \lambda, u_{\max} = 1$

Given : Reward function $R(s, z)$

Initialize replay buffer $\mathcal{D} = \varnothing$, policy π_{θ_u}

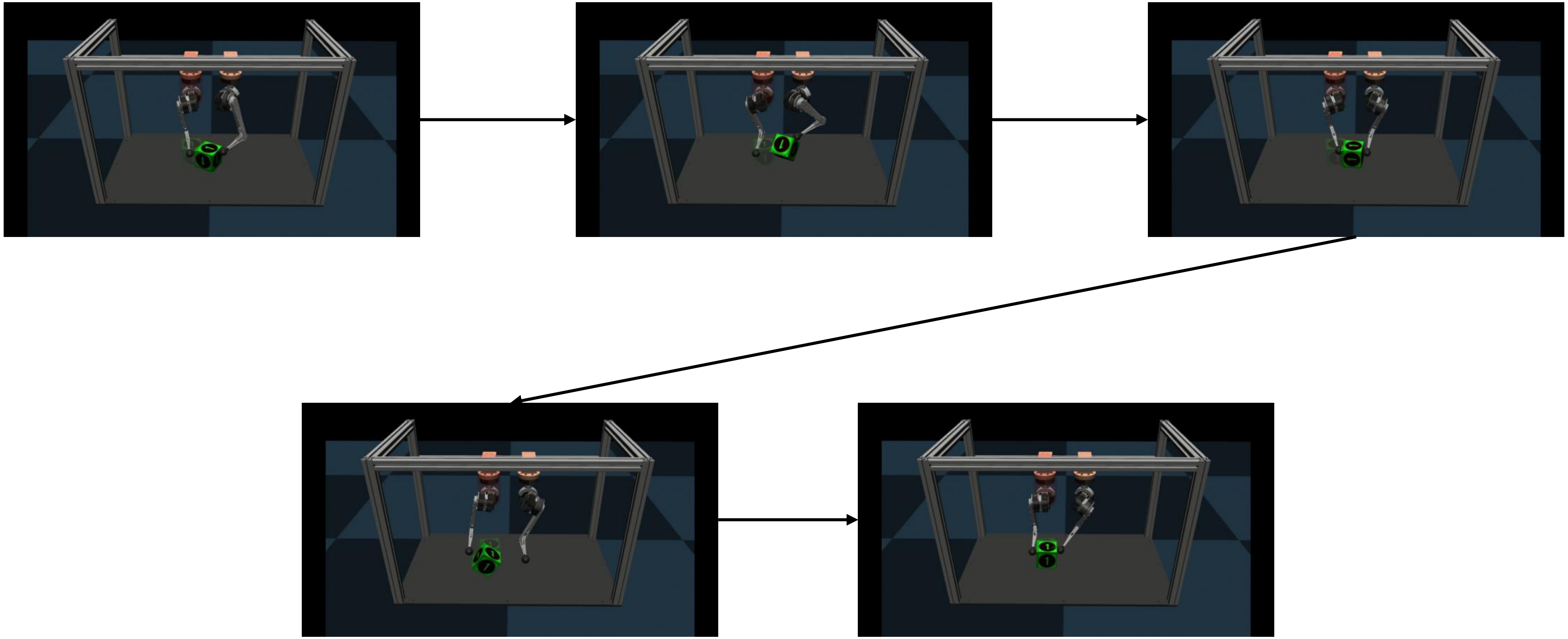
while *not done* **do**

- `// Sample from offline dataset`
- Sample H consecutive trajectories, $(\tau^k, s_g^k, s_i^k)_{k=1}^H \sim \mathcal{D}_{\tau,W}$
- Sample $z^k \sim q_\phi(\cdot|\tau^k, s_g^k s_i^k)$, for $k = 1, \cdots, H$
- Compute inverse mapping $u^k = h_\psi^{-1}(z^k; s_i^k)$ and reward $R^k = R(s_i^k, z^k)$, for $k = 1, \cdots, H$
- $\mathcal{D} \leftarrow \mathcal{D} \cup (s_i^k, s_g^k, u^k, R^k)_{k=1}^H$
- **if** *goal-conditioned* **then**
  - $\mathcal{D} \leftarrow \mathcal{D} \cup \mathrm{HER}((s_i^k, s_g^k, u^k, R^k)_{k=1}^H)$
- **end**
- `// Sample model-based rollouts`
- $\bar{s}_i^1 = s_i^1$
- **for** $t = 0 : N_m - 1$ **do**
  - Sample base skill, $\bar{u}^t \sim \pi_{\theta_u}(\bar{s}_i^t)$
  - Clamp $\bar{u}^t \leftarrow u_{\max} \cdot \mathrm{Tanh}(\bar{u}^t)$
  - Compute forward mapping $\bar{z}^t = h_\psi(\bar{u}^t; \bar{s}_i^t)$
  - Predict next state $\bar{s}_i^{t+1} = f_{\theta_f}(\bar{s}_i^t, \bar{z}^t)$, and evaluate reward $\bar{R}^t = R(\bar{s}_i^t, \bar{z}^t)$
- **end**
- $\mathcal{D} \leftarrow \mathcal{D} \cup (\bar{s}_i^k, \bar{s}_g^k, \bar{u}^k, \bar{R}^k)_{k=1}^{N_m}$
- **if** *goal-conditioned* **then**
  - $\mathcal{D} \leftarrow \mathcal{D} \cup \mathrm{HER}((\bar{s}_i^k, \bar{s}_g^k, \bar{u}^k, \bar{R}^k)_{k=1}^{N_m})$
- **end**
- Update $\pi_{\theta_u} \leftarrow \mathrm{SAC}(\theta_u, \mathcal{D})$

end

Output: π_{θ_u}

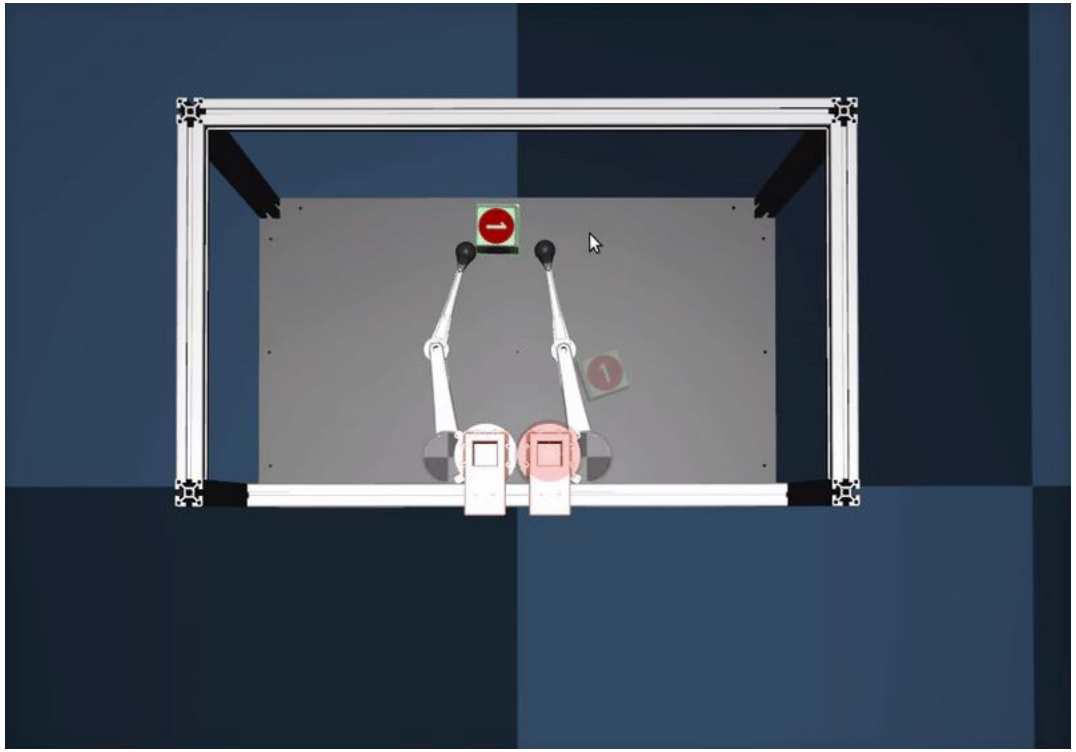
Model-based offline RL

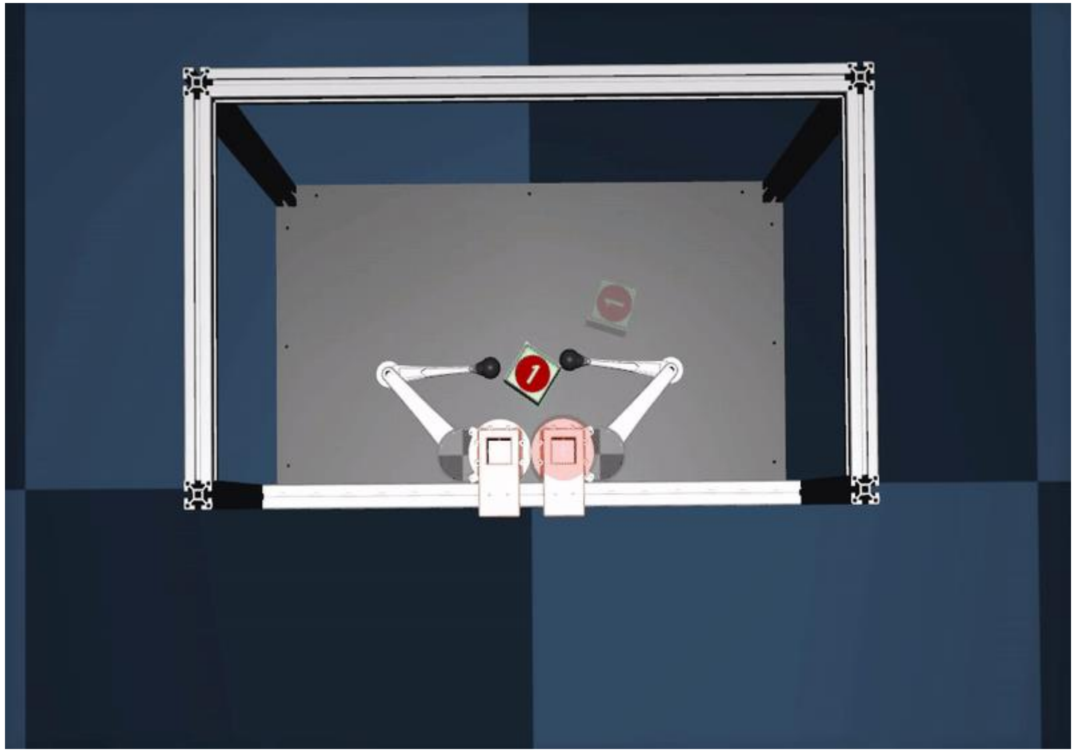


4. Result

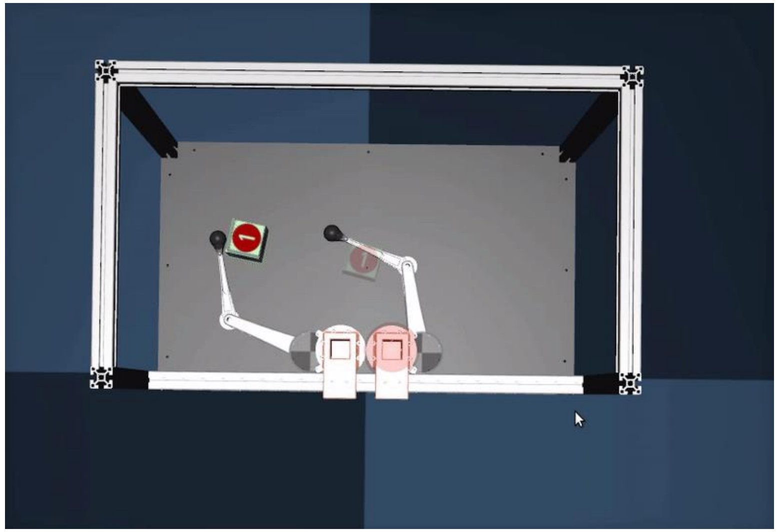
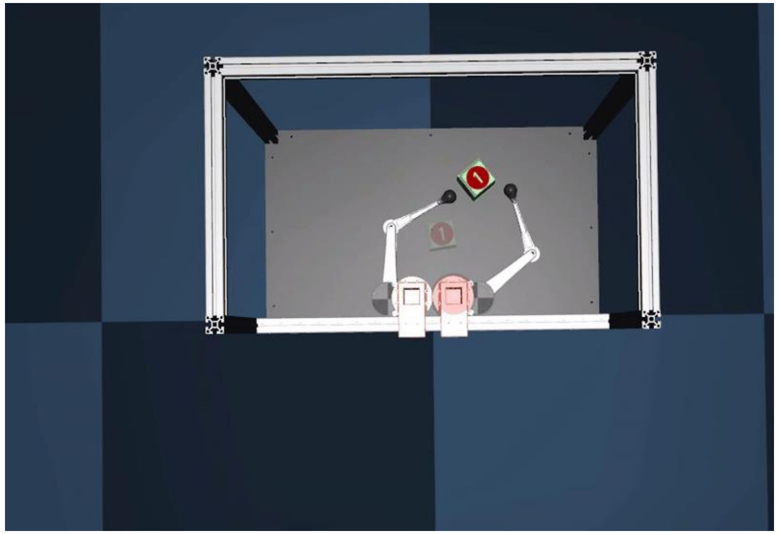
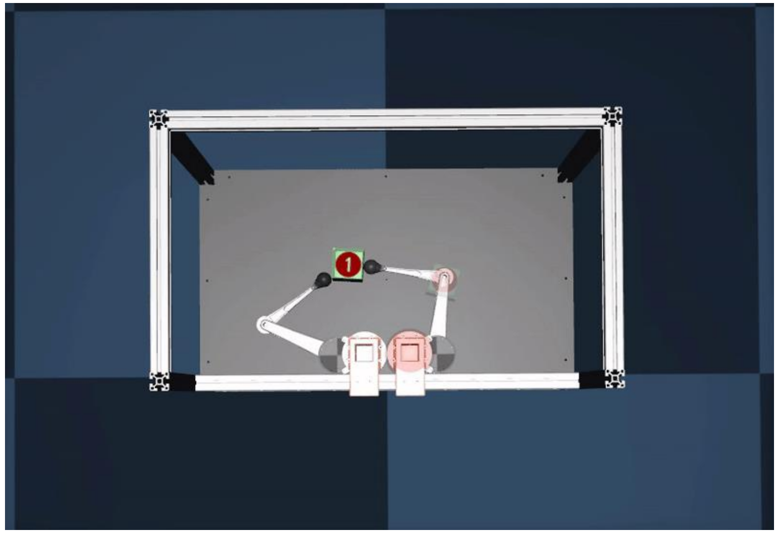
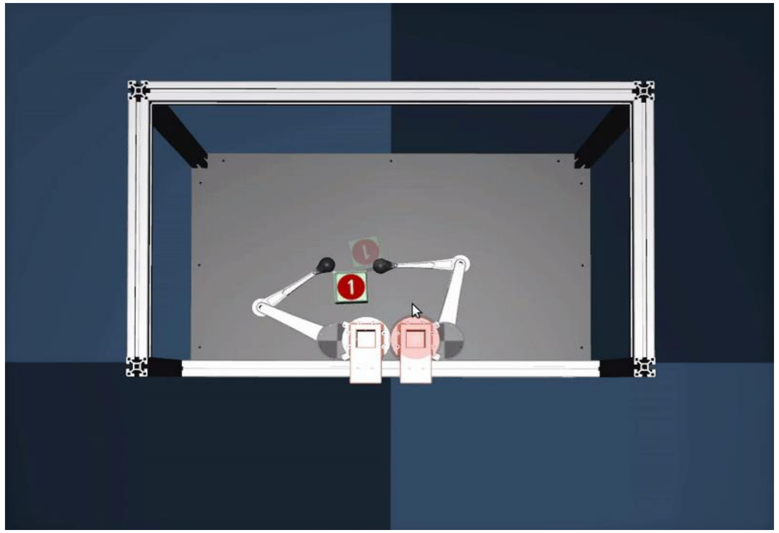
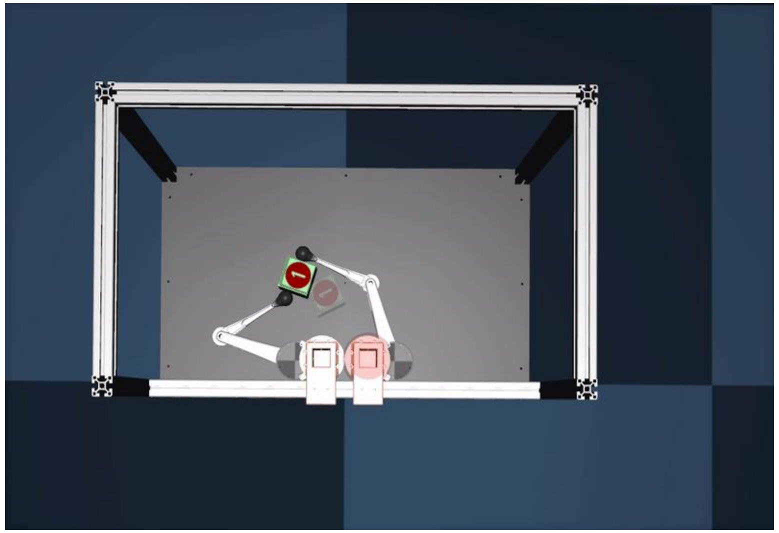
Simulation results

Simulation results (2D)

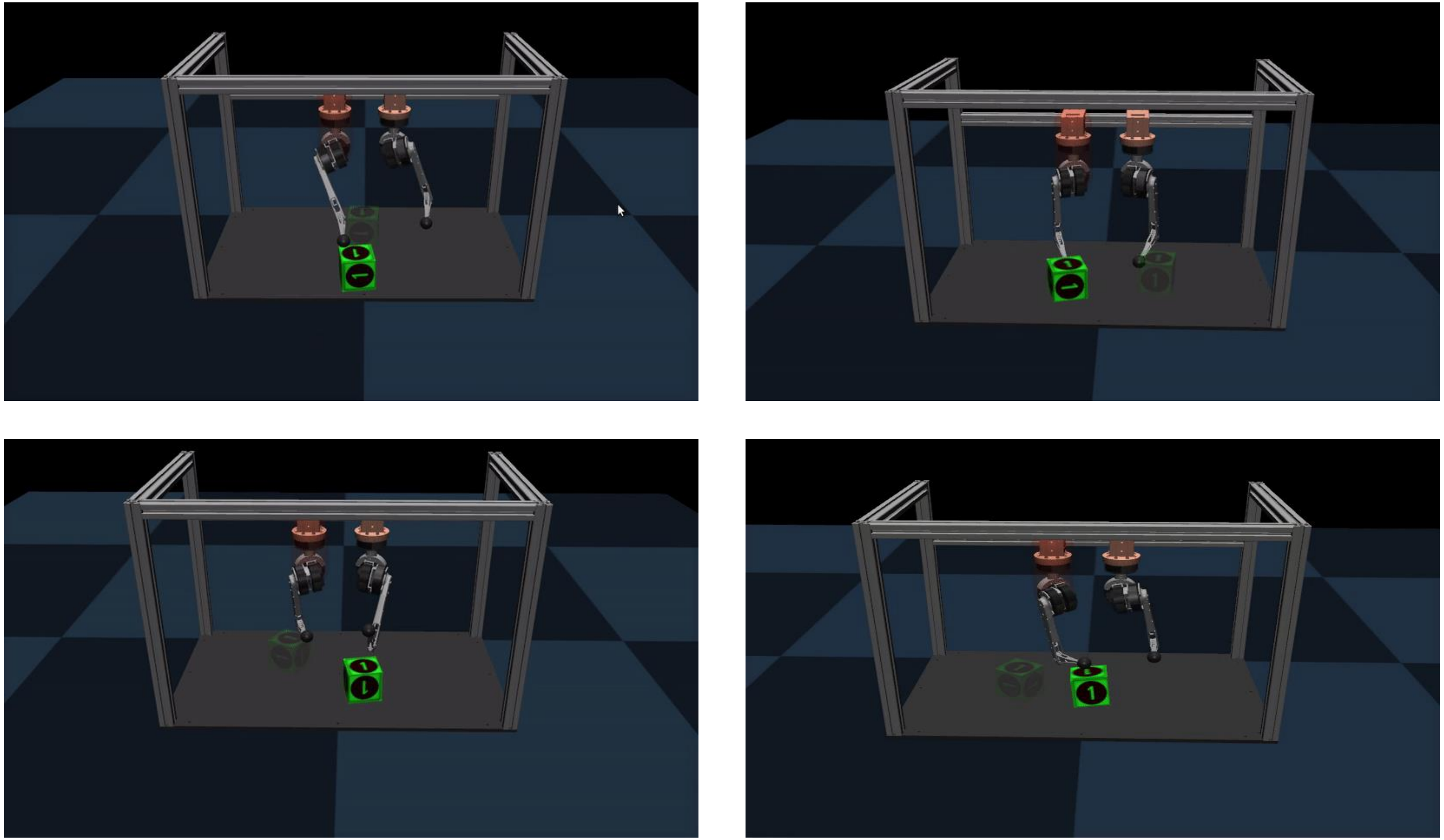




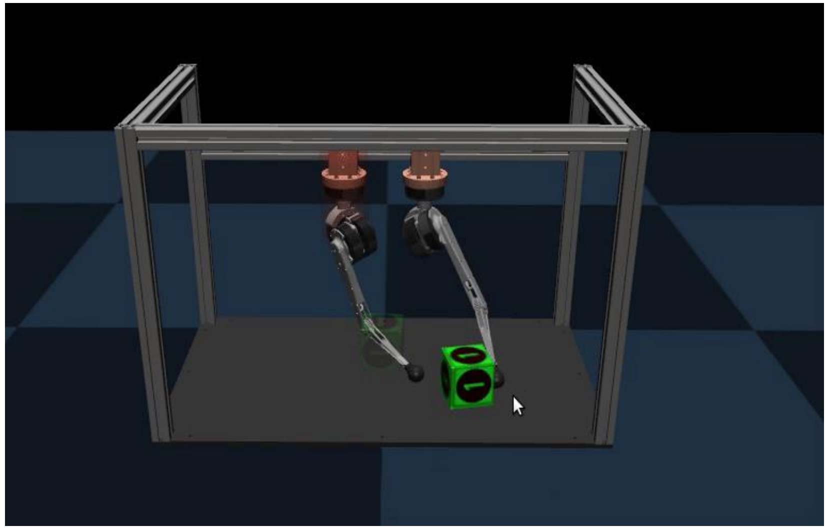
Simulation results (2D)

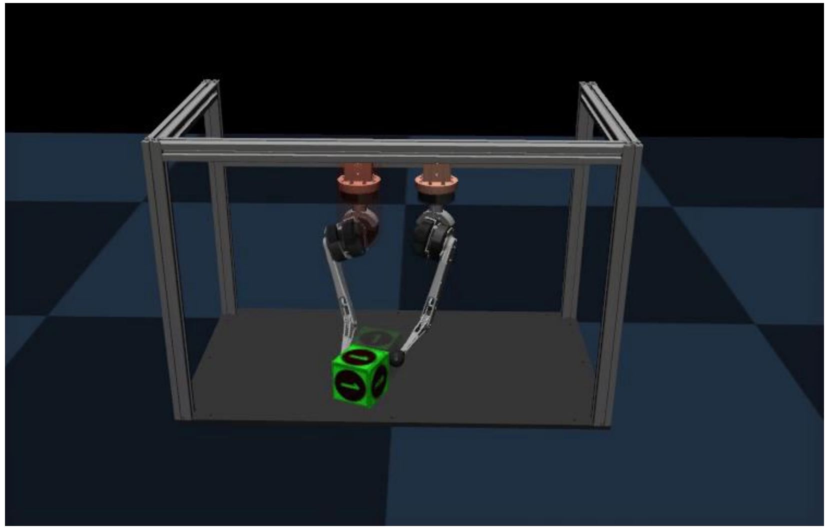






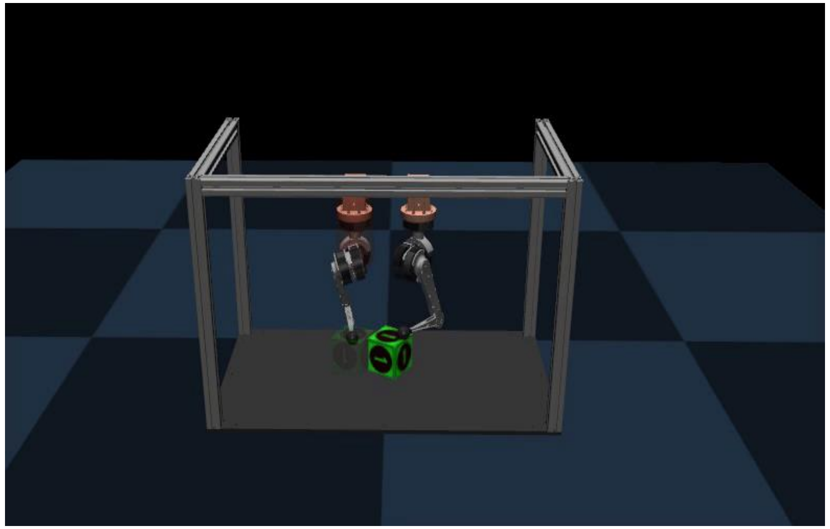
Simulation results (3D)



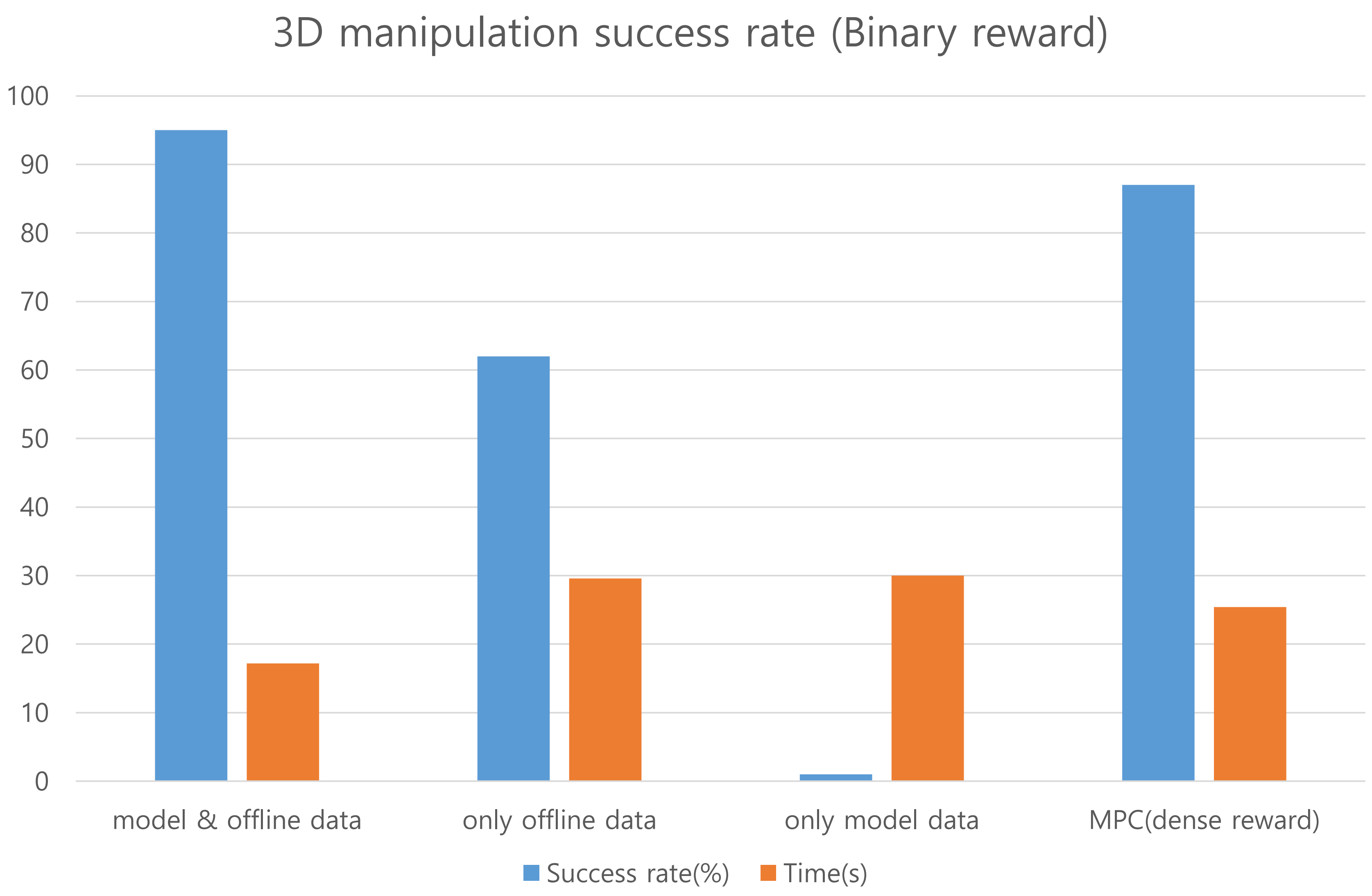
Simulation results (3D)







Simulation results (3D)



Simulation results (3D)

	Offline RL	MPC
2D manipulation	0.0008s	0.08s
3D manipulation	0.001s	0.1s

Q&A

Thank you

NAVER LABS

SAC Hyper-parameter

Parameter	Value
Replay_size	1e7
Gamma	0.96
Polyak	0.995
Policy learning rate	3e-5
Q function learning rate	3e-4
Alpha	1.0
Batch size	256
Gradient iterations per one step (update_every)	40
Policy hidden units	512
Number of Policy layers	2