

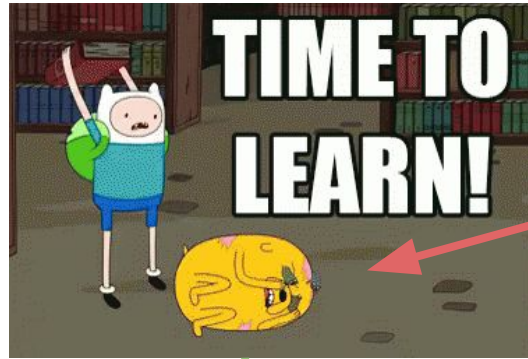


Fast skill acquisition with goal-conditioned RL

Utilising behavioural priors and self-supervision

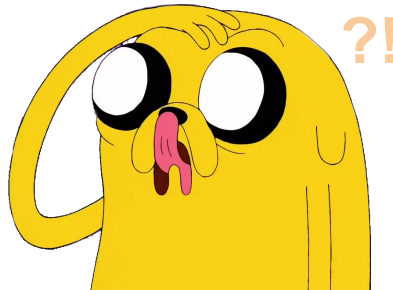
Todor Davchev

Goal-conditioned Skill Learning



No goal

Goal



does goal-conditioning help robots?

Variability in skill

Counteract these factors of variation using structure!

Perceptual Variations

scene



light



shadow



noise

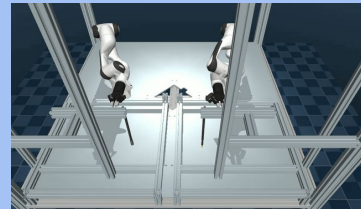


Dynamical Variations

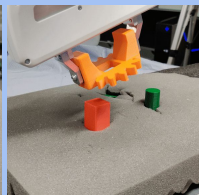
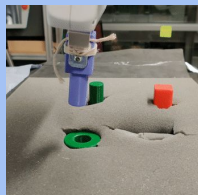
geometries



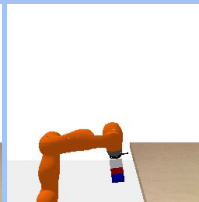
vary configuration



mass, friction



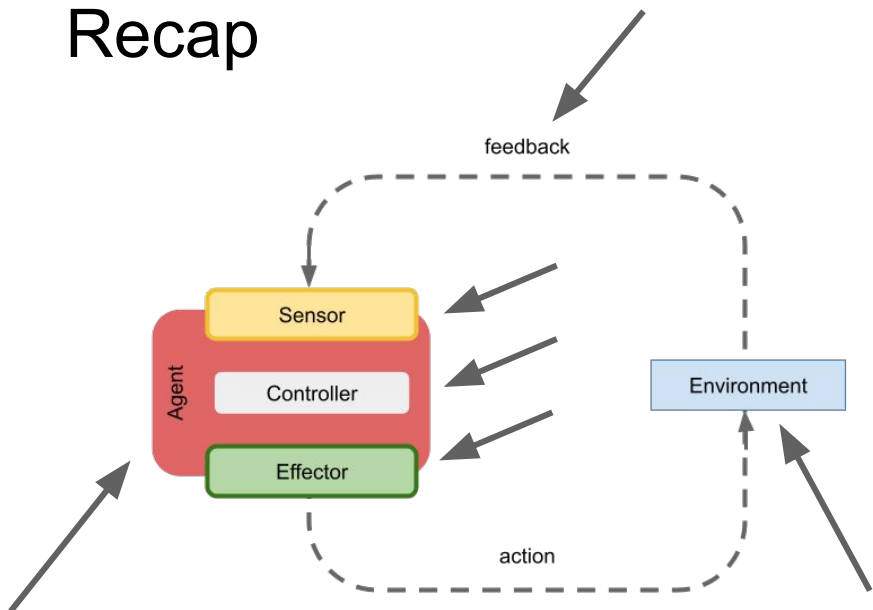
speed of execution



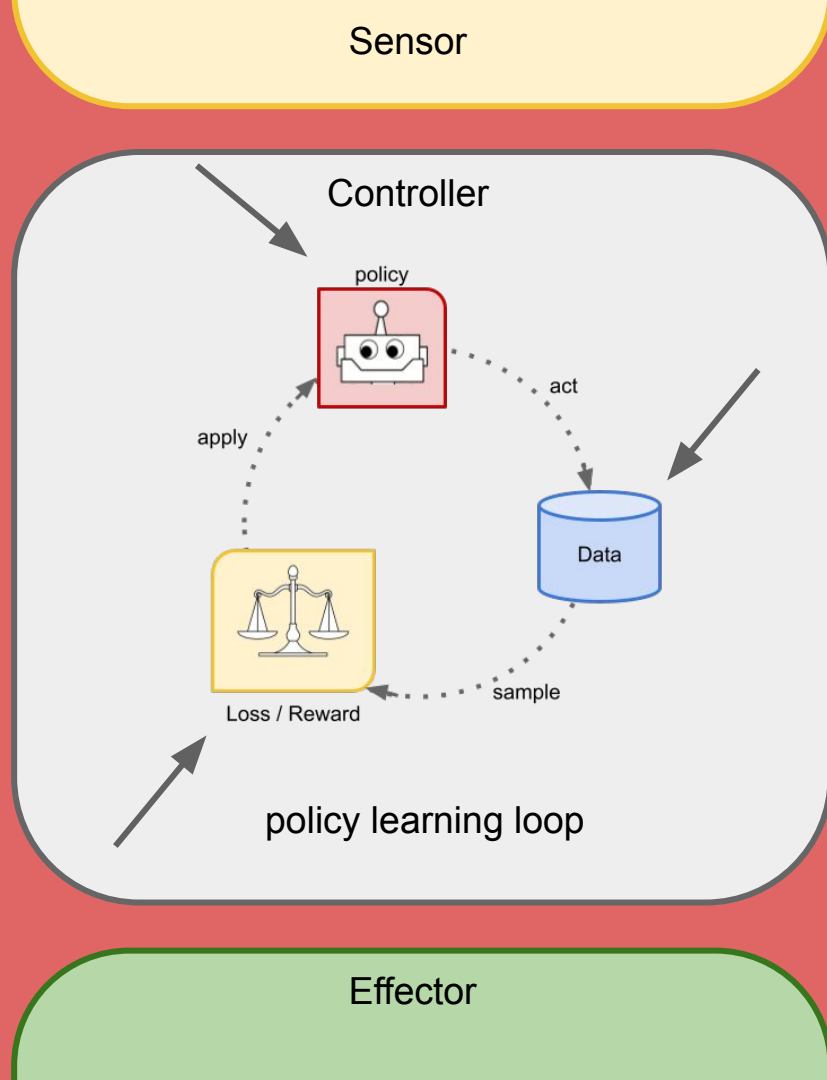
goal location



Recap



a robot acting loop



policy learning loop

Overview

1. Goal-conditioned Policy Learning (goals as input to policy)

- Residual Learning from Demonstration: Adapting DMPs for Contact-rich Manipulation, RA:L and ICRA 2022
- Wish you were here: Hindsight Goal Selection for Long-Horizon Dexterous Manipulation, ICLR 2022

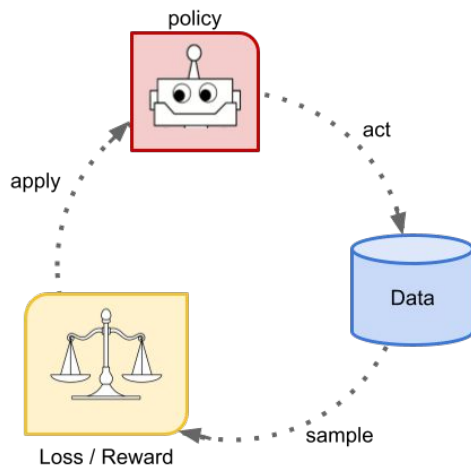
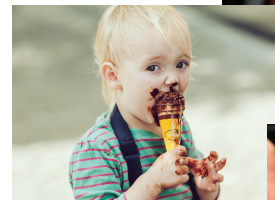
2. Goal-conditioned Reward Learning (goals as input to rewards)

- Model-Based Inverse Reinforcement Learning from Visual Demonstrations, CoRL 2020
- Learning Time-Invariant Reward Functions through Model-Based Inverse Reinforcement Learning, Under review

3. Other directions (if time permits)

- Learning Structured Representations of Spatial and Interactive Dynamics for Trajectory Prediction in Crowded Scenes, RA-L 2021
- An Empirical Evaluation of Adversarial Robustness under Transfer Learning, ICML 2019, Workshop on Improving Generalization

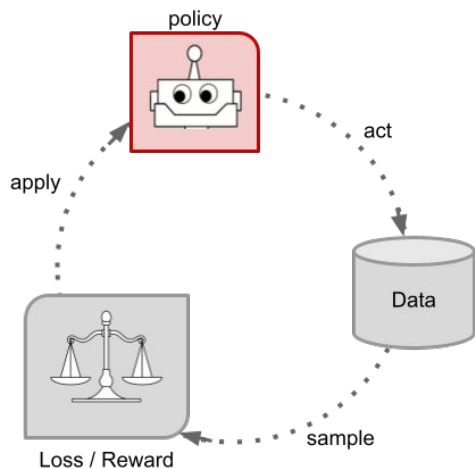
Robot Learning for Contact-rich Manipulation



Objective:

- adapt knowledge to environmental changes
- account for model imperfections;
- long-term: enable sample-efficient lifelong learning of manipulation-based skills that can scale to complex sequential settings.

Residual Learning from Demonstration



Objective:

- Robust and safe policies
- Sample efficient learning
- relax the need for model re-training in new environments
- enable fast adaptation to unseen tasks

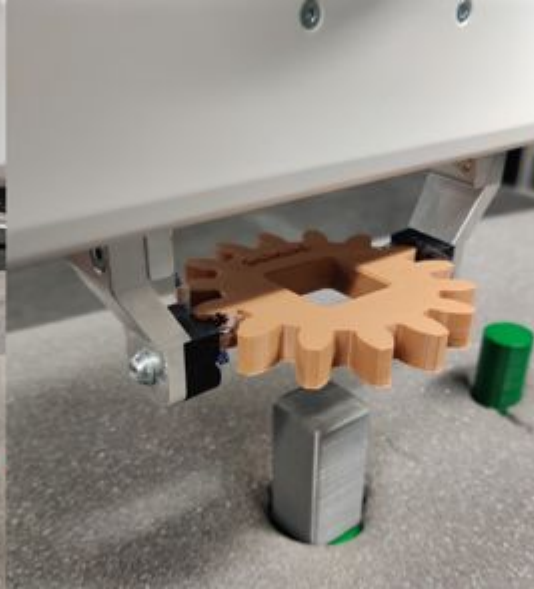
Tasks

Peg Insertion



- dependent on position of end effector

Gear Insertion



- requires precise orientation
- strong downwards push

RJ-45 Insertion



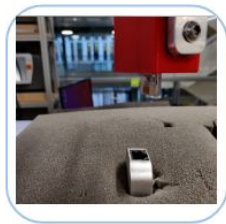
- requires precise position and orientation
- refined force control

Learn from Demonstration

1. Record a Demonstration
2. Extract a policy
3. Execute from any start/goal



Learning from Demonstration



t_0

...



t_m

...

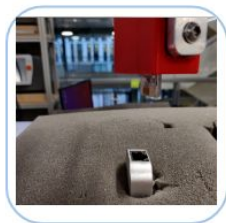


t_n

Learning from Demonstration

Base Policy

100 Hz



t_0

...



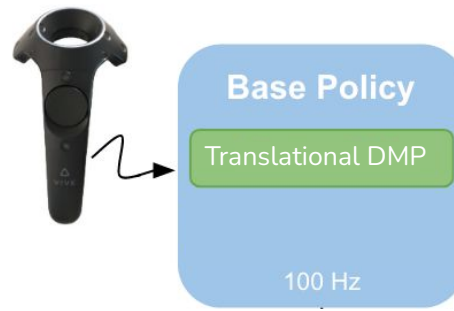
t_m

...



t_n

Learning from Demonstration



t_0

...



t_m

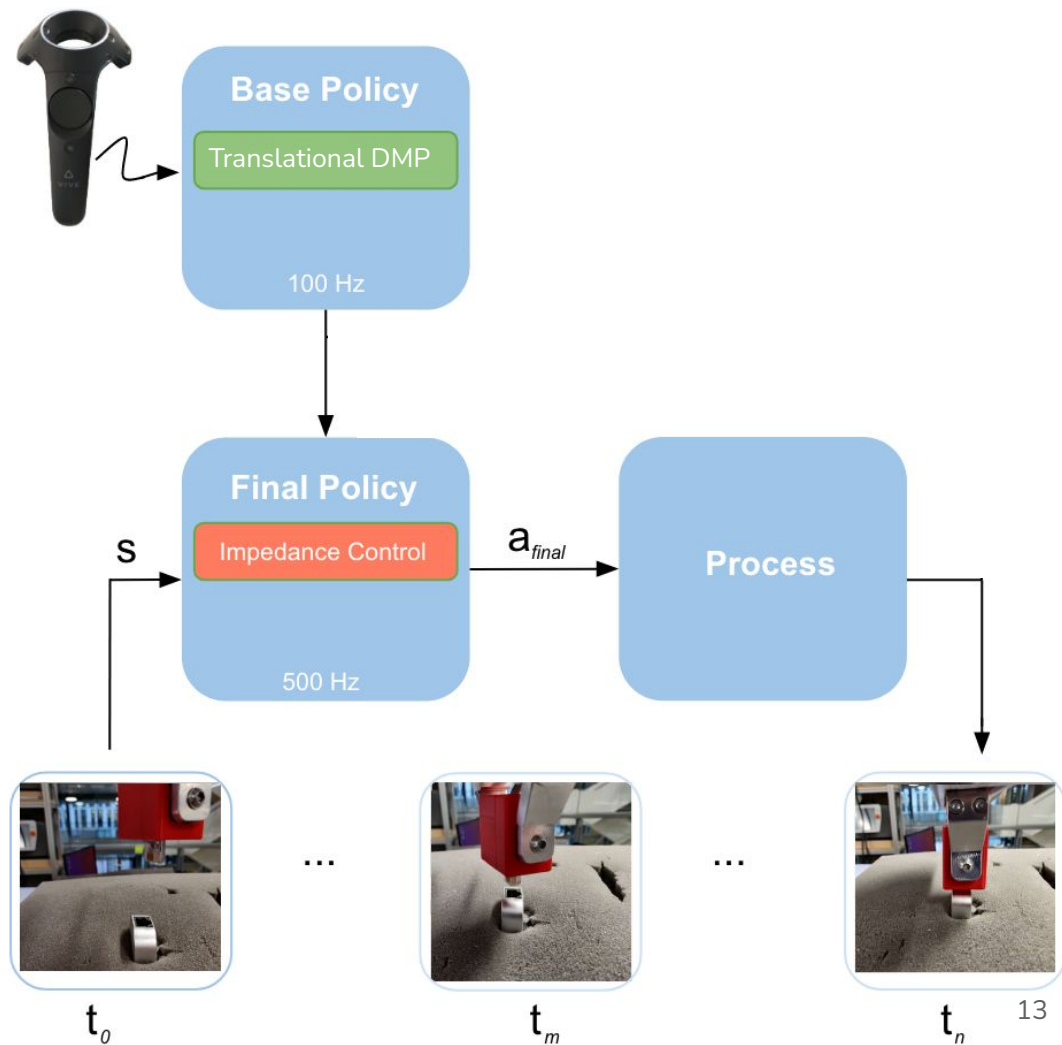
...



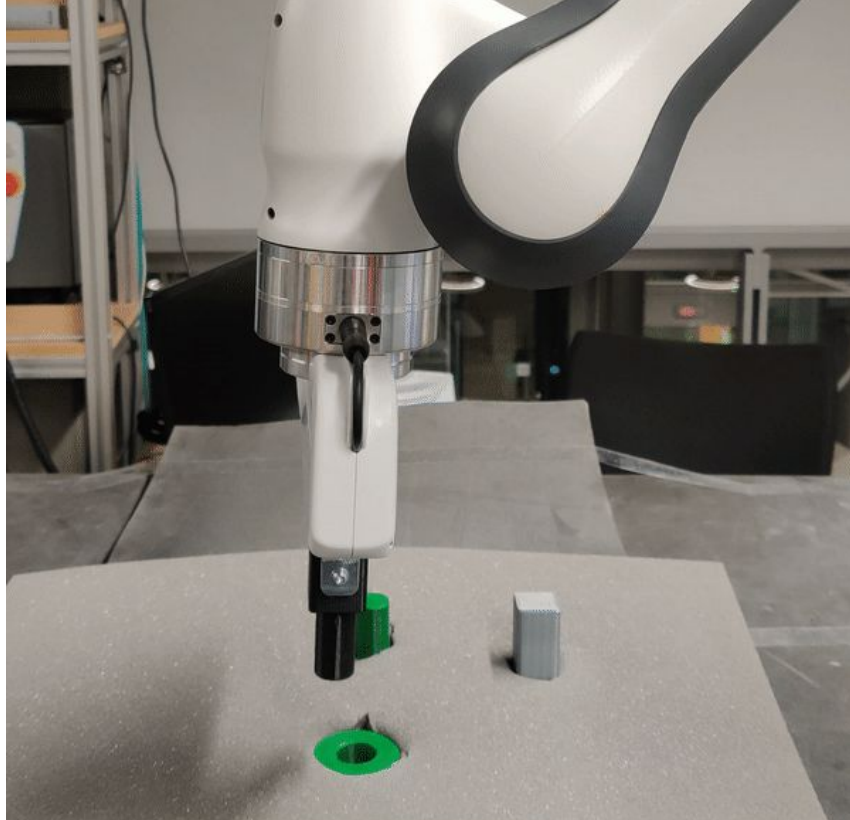
t_n

Ijspeert, Auke Jan, et al. "Dynamical movement primitives: learning attractor models for motor behaviors." *Neural computation* 25.2 (2013): 328-373.

Learning from Demonstration



Rote Imitation

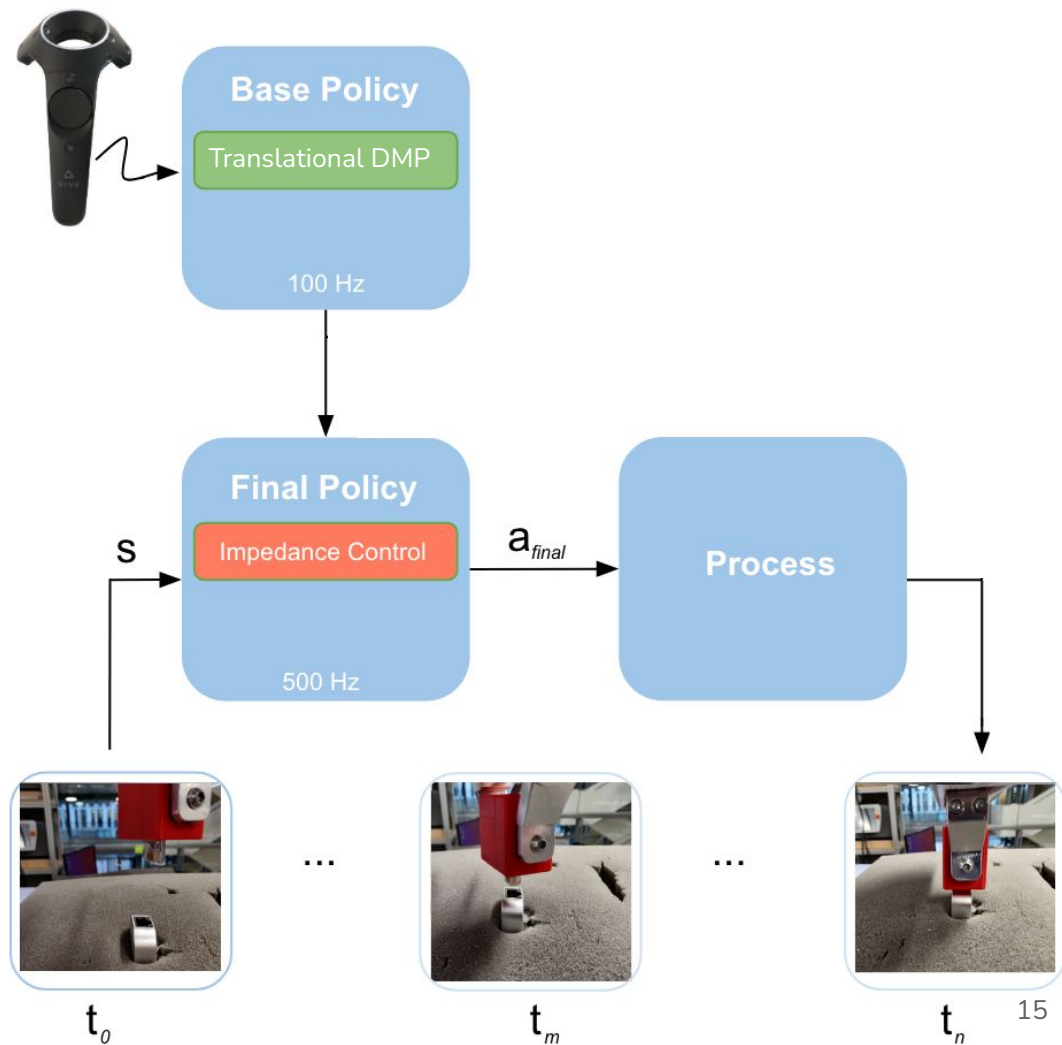


Small changes can be fatal

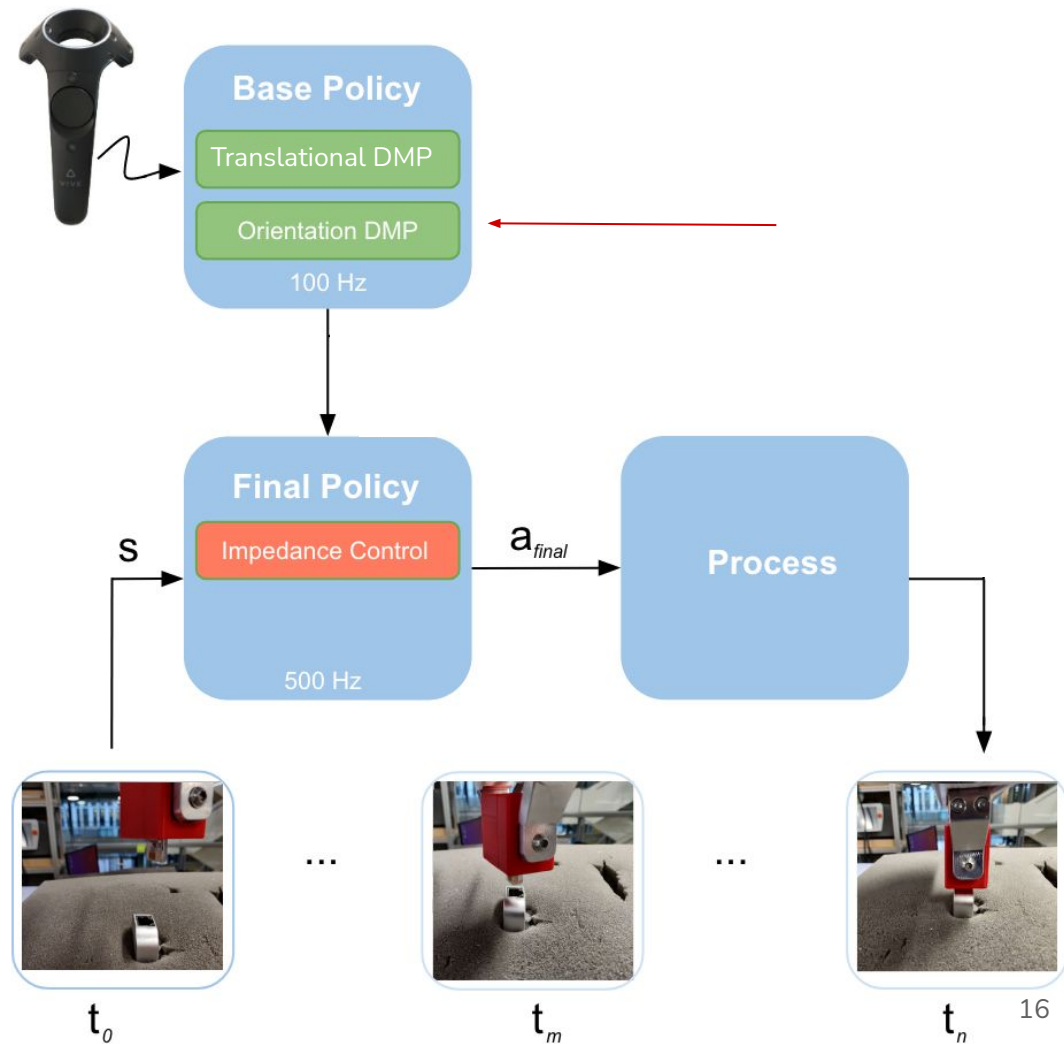
Model the world is hard



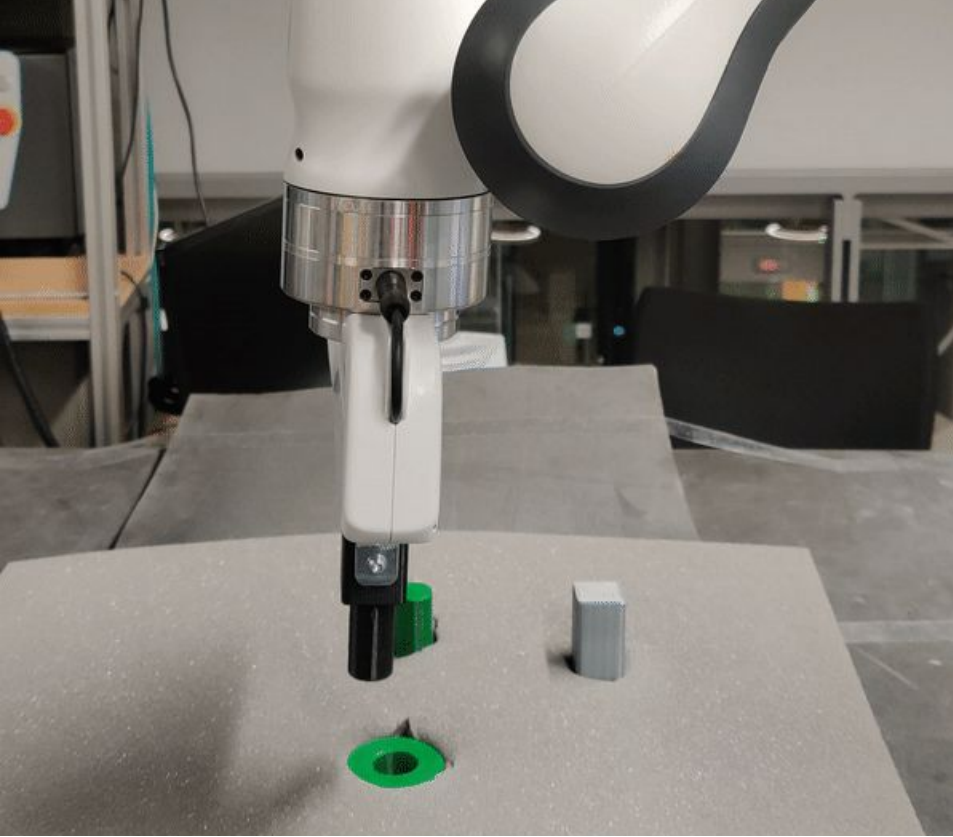
Learning from Demonstration



Learning from Demonstration

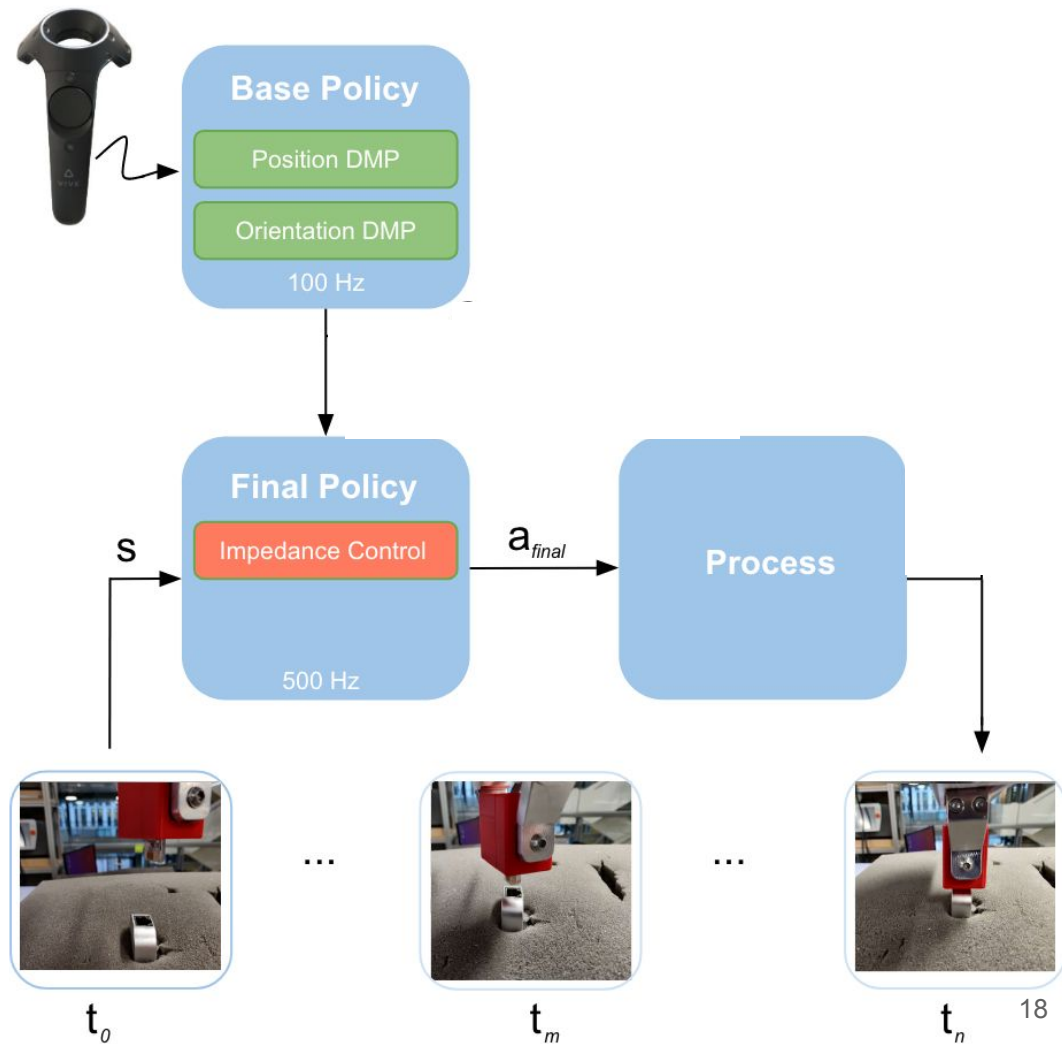


Ude, A., Nemec, B., Petrić, T. and Morimoto, J., 2014, May. Orientation in cartesian space dynamic movement primitives. In *2014 IEEE International Conference on Robotics and Automation (ICRA)* (pp. 2997-3004).



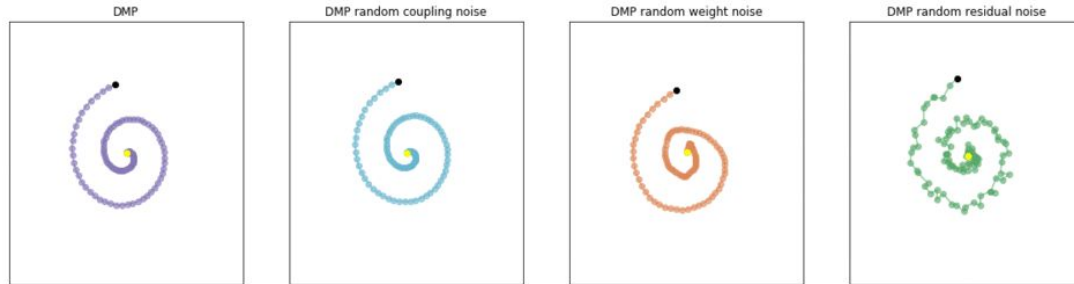
Better but not ideal

Residual Learning from Demonstration



T. Davchev, K. Luck, M. Burke, F. Meier, S. Schaal, S. Ramamoorthy, "Residual Learning from Demonstration: Adapting DMPs for Contact-rich Manipulation," in IEEE Robotics and Automation Letters

Adapting the DMP Formulation



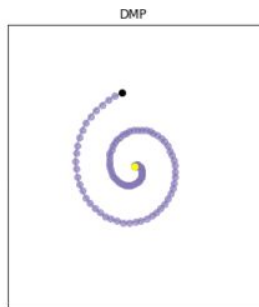
$$\dot{y} = \frac{1}{\tau^2}(\alpha_v(\beta_v(g - x) - \tau y) + z f_{\omega+\eta} + C_t(\eta)) + \dot{\eta}$$

Correcting the DMP formulation

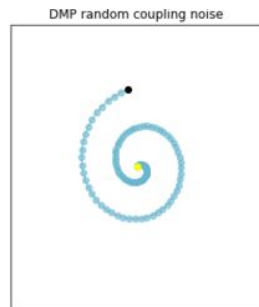
Adapting DMPs

Table 4.1: Accuracy of applying exploration during learning on different parts of the DMP formulation (Type colour from Fig. 5.1). Eff. is efficiency- the number of episodes a model was trained for.

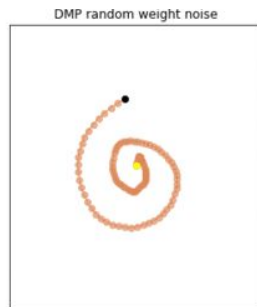
Type	Exploration Type	Model	Easy	Hard	Average	Eff.	Reward
A	No corrections	DMP (Ijspeert et al., 2013)	24.0% \pm 2.5	8.0% \pm 1.4	16.0% \pm 2.0	n/a	n/a
B	phase modulated coupling	eNAC (Peters & Schaal, 2007)	7.2% \pm 1.9	3.4% \pm 2.2	5.3% \pm 2.1	8K	$\exp\{-L_1\}$
C	forcing-term parameters	FDG (Peters & Schaal, 2008)	23.6% \pm 4.3	16.2% \pm 1.9	19.9% \pm 3.1	8K	$\exp\{-L_1\}$
C	forcing-term parameters	PoWER (Kober & Peters, 2009)	32.2% \pm 2.8	14.4% \pm 2.7	23.3% \pm 2.8	8K	$\exp\{-L_1\}$
D	task space translation	rLfD (ours)	94.8% \pm 1.3	55.0% \pm 2.7	74.9% \pm 2.0	700	$\mathbb{1}[L_2 \leq \kappa]$



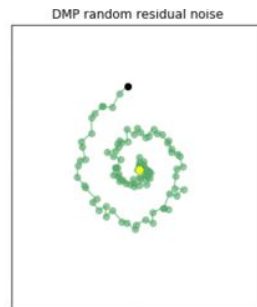
A



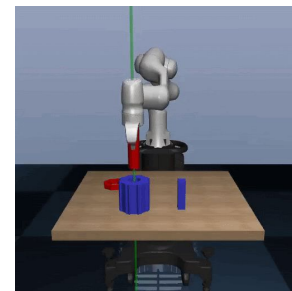
B



C



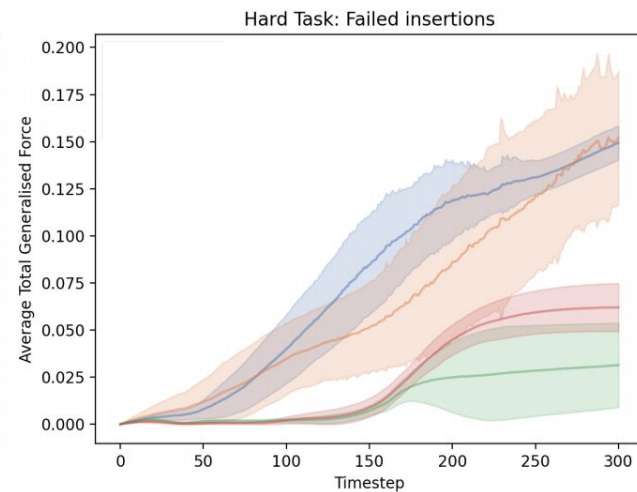
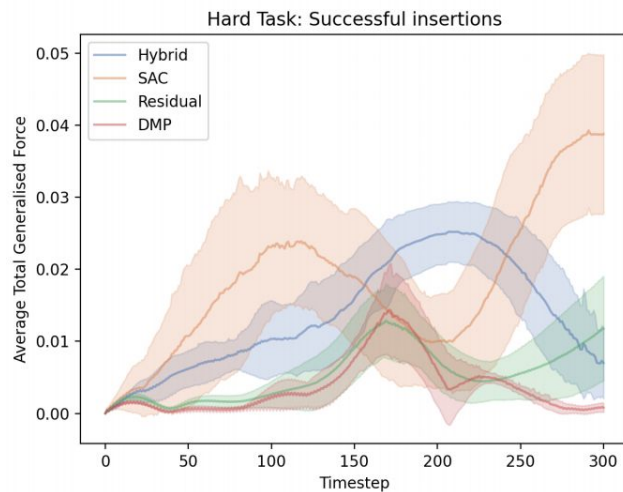
D



task

Gentle to the joints?

How much force on average would all joints experience?



Range of Considered Tasks

Physical Tasks

Peg insertion

- Challenging World
- External Biases

Gear insertion

- Orientation dependent
- Tighter Hole

Lan Cable Insertion

- Smaller Hole
- Force Dependent



Full pose corrections in the real world

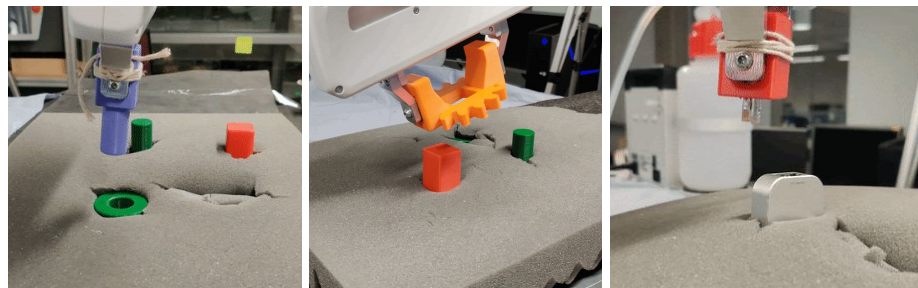
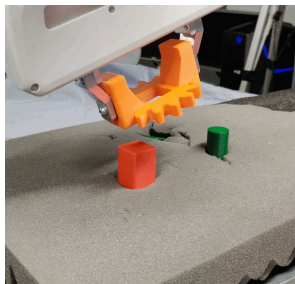


Table 4.5: Utilising nonlinear policy accuracy comparison. Pose baselines are named as: translation policy / orientation policy.

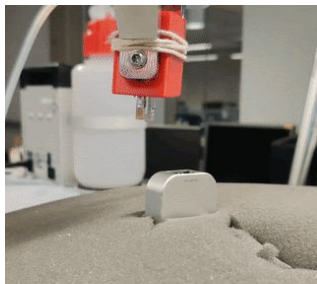
Type	Corrections	Adaptive Policy	Peg	Gear	RJ-45	Average
A	No corrections	None / None	52.6% \pm 0.7	41.5% \pm 1.5	0.0% \pm 0.0	31.4% \pm 0.7
A	translation	Linear / None	80.7% \pm 4.0	58.7% \pm 1.1	14.6% \pm 1.6	51.3% \pm 2.2
D	translation	PPO / None	94.2% \pm 0.9	50.0% \pm 0.4	57.2% \pm 2.1	67.4% \pm 1.1
A	full pose	Linear / Random (ours)	60.3% \pm 2.9	76.2% \pm 2.6	57.3% \pm 2.5	64.6% \pm 2.7
A	full pose	Random / Random (ours)	91.0% \pm 1.9	86.9% \pm 1.7	64.8% \pm 1.2	80.9% \pm 1.6
D	full pose	PPO / PPO (ours)	97.9% \pm 1.2	92.2% \pm 2.6	70.6% \pm 1.4	86.9% \pm 1.7

Transfer residual policies across tasks

Gear



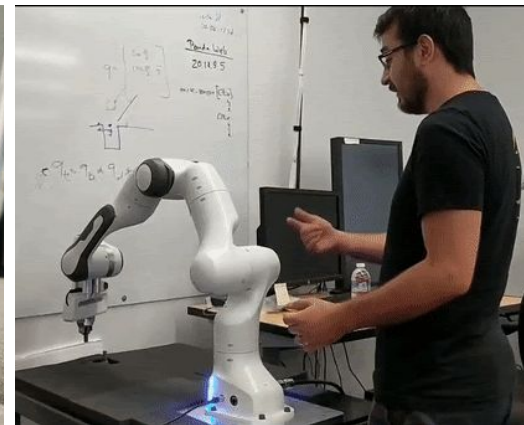
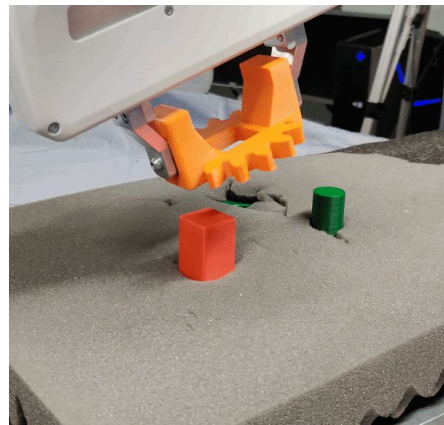
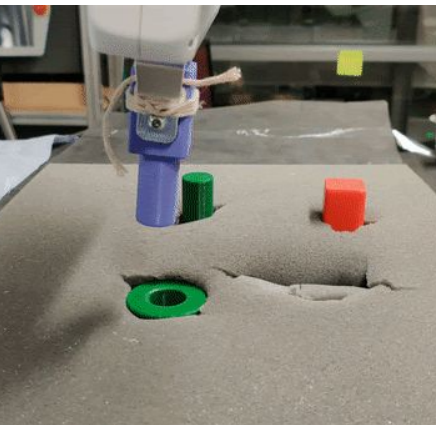
RJ-45



rLFD allows for successful policy transfer on both tasks,
requires 8 times less training!

Table 4.7: Successful insertions on transfer, comparison. Pose baselines are named as: translation policy / orientation policy.

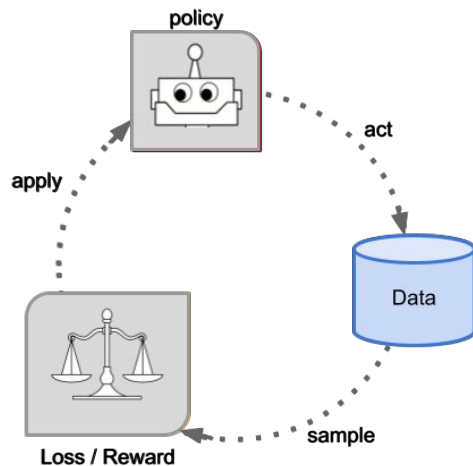
Type	Corrections	Adaptive Policy	Gear	RJ-45	Average	Eff.
D	full pose	(full training) π_{targ}	92.2% \pm 2.6	70.6% \pm 1.4	81.4% \pm 2.0	500
D	full pose	(full training) π_{src}	85.4% \pm 1.4	54.5% \pm 3.2	69.9% \pm 2.3	500
D	full pose	(3-shot) π_{targ}	70.3% \pm 4.0	59.1% \pm 3.1	64.7% \pm 3.6	60
D	full pose	(3-shot) $\pi_{src \rightarrow targ}$	92.0% \pm 2.1	70.6% \pm 1.7	81.3% \pm 1.9	60



Residual Learning from Demonstration (Key observations)

- a framework for residual full pose corrections
- more robust and sample efficient than adapting DMP parameters directly;
- adapt to world changes efficiently and online;
- a less forceful solution than alternative schemes;
- Transfer residual policy across different tasks.

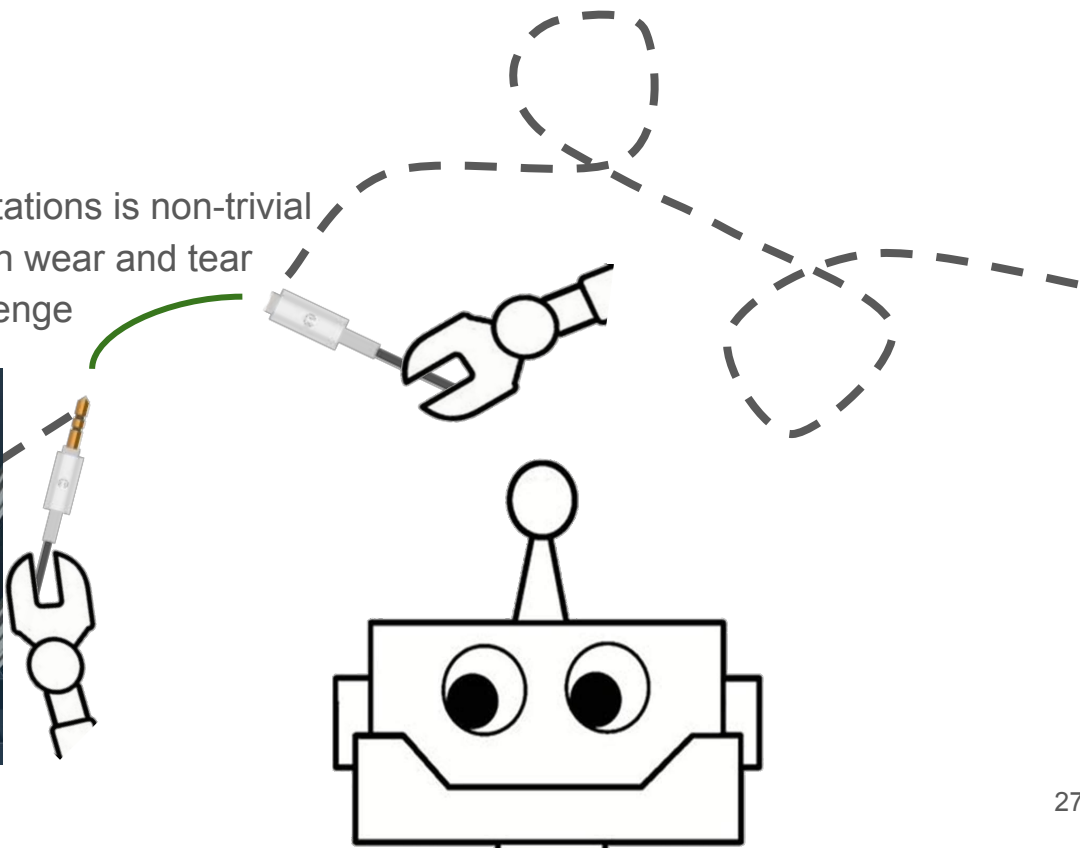
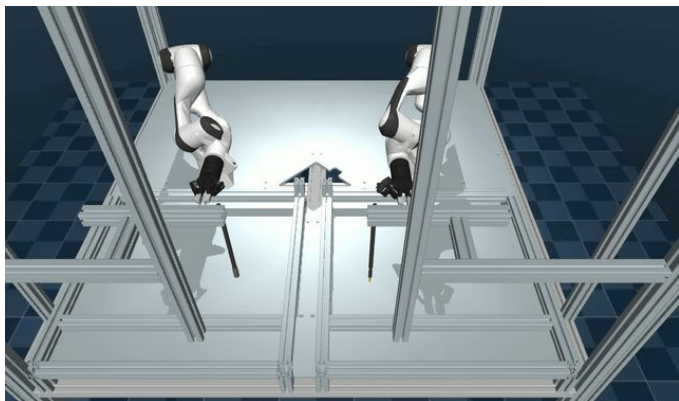
Self-supervised RL



- handle long-horizon complex sequential tasks
- scale to harder contact-rich tasks
- maintain few-shot properties remain sample efficient;

Solving Complex Sequential Tasks with Sparse Rewards can be Hard!

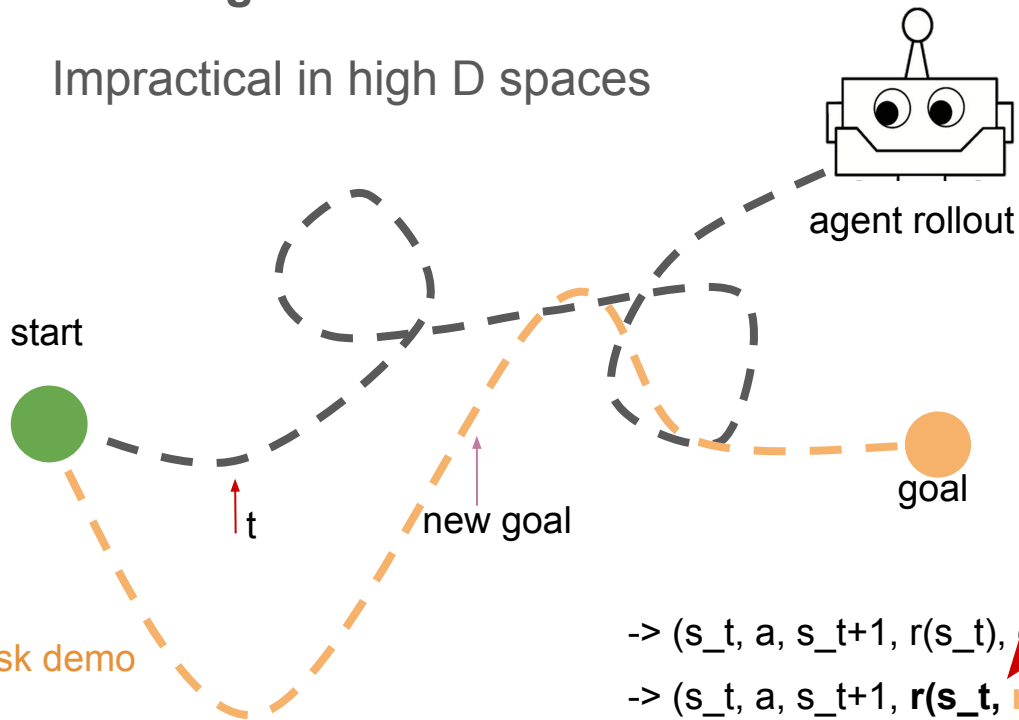
- # narrow passage phases
- Reduced access to positive signal
- Extracting useful state/goal representations is non-trivial
- Larger sample requirements results in wear and tear
- Broadly, an exciting exploration challenge



Dealing with redundant data: task-constrained distribution

Learning from raw states

- Impractical in high D spaces



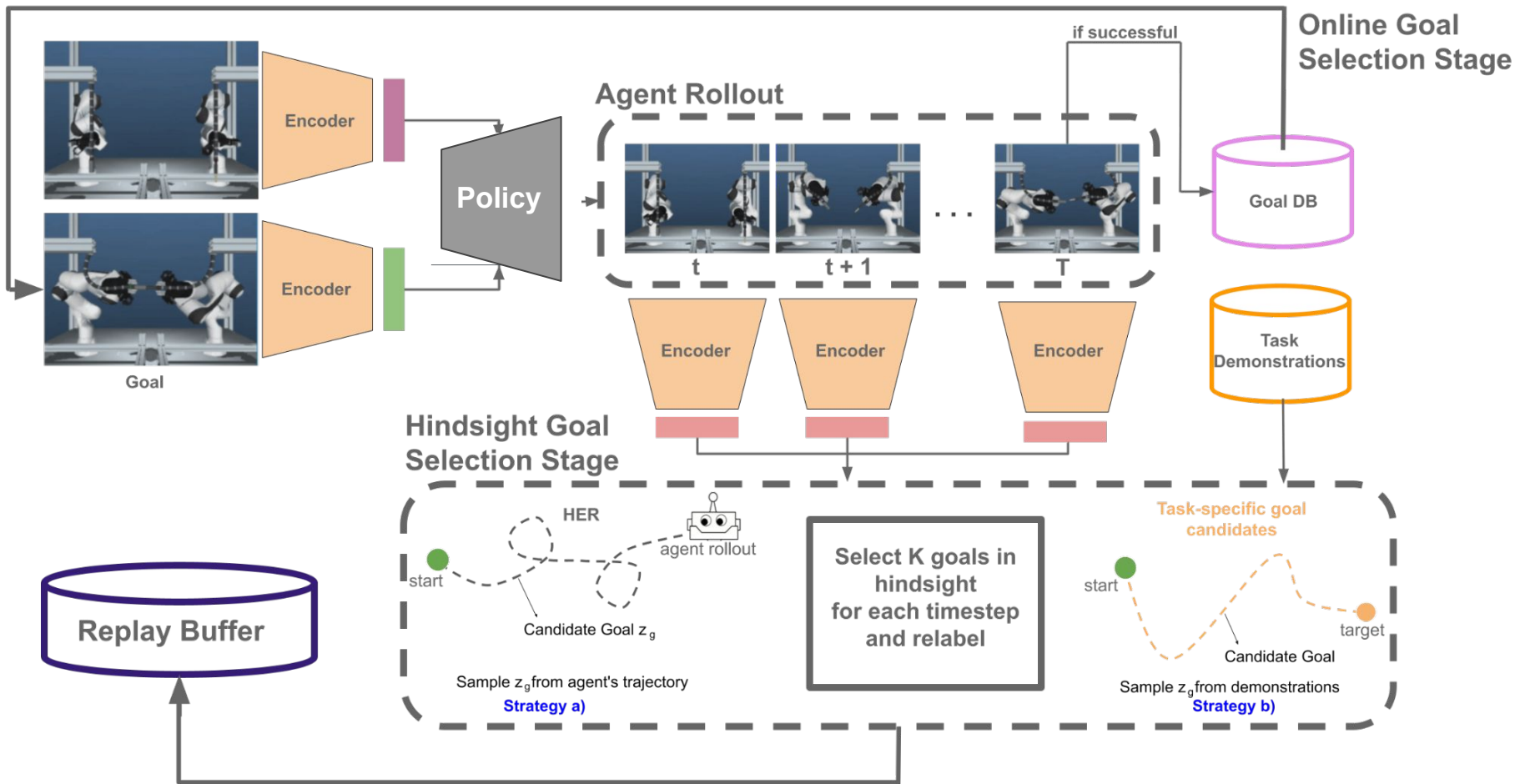
Instead use encoder ψ

- Encode to low dimensional latent space
- Explicitly encode a notion of progress
- Utilise engineered and learnt time-consistent representations

-> $(s_t, a, s_{t+1}, r(s_t), \text{goal})$

-> $(s_t, a, s_{t+1}, r(s_t, \text{new goal}), \text{new goal})$

Hindsight Goal Selection for Demo-Driven RL (HinDRL)

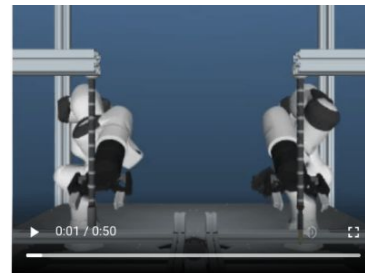


Applying HinDRL on long-horizon sequential tasks

Environment	BC	DPGfD	HER (final)	HER (future)	HinDRL (Our)
	Mean (Std)	Mean (Std)	Mean (Std)	Mean (Std)	Mean (Std)
Parameterised Reach (2wp)	82.05% (0.0)	88.62% (7.7)	84.92% (28.6)	86.92 (5.2)	92.61% (4.7)
Bring Near	88.90% (0.0)	99.74% (1.0)	32.03% (41.8)	97.75% (2.9)	98.79% (4.1)
Bring Near + Orient	57.89% (0.0)	83.52% (18.0)	8.33% (27.6)	23.28% (33.5)	90.77% (10.0)
Bimanual Insertion	16.06% (0.0)	4.28% (4.4)	3.38% (3.3)	18.39% (12.3)	78.92% (10.3)
Average	61.23% (0.0)	69.04% (7.8)	32.17% (25.32)	56.59% (12.9)	90.27% (7.3)

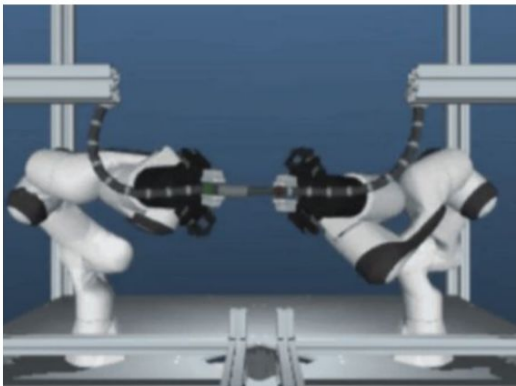
Table 1: Records performance in terms of accuracy.

* Using an encoder from engineered features



Few-shot Task Performance: Dependency on demonstrations

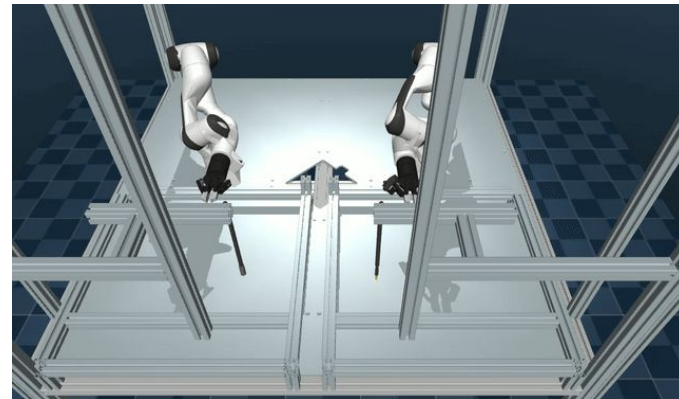
Full Insertion



Environment	BC Mean (Std)	DPGfD Mean (Std)	HER (future) Mean (Std)	HinDRL (Our) Mean (Std)
55 demos	16.06% (0.0)	4.28% (4.4)	18.39%	78.92% (10.3)
10 demos	2.94% (0.0)	0.0% (0.0)	0.0% (0.0)	32.05% (23.3)
5 demos	0.52% (0.0)	0.0% (0.0)	0.0% (0.0)	17.78% (18.9)
1 demos	0.13% (0.0)	0.0% (0.0)	0.0% (0.0)	12.58% (18.3)

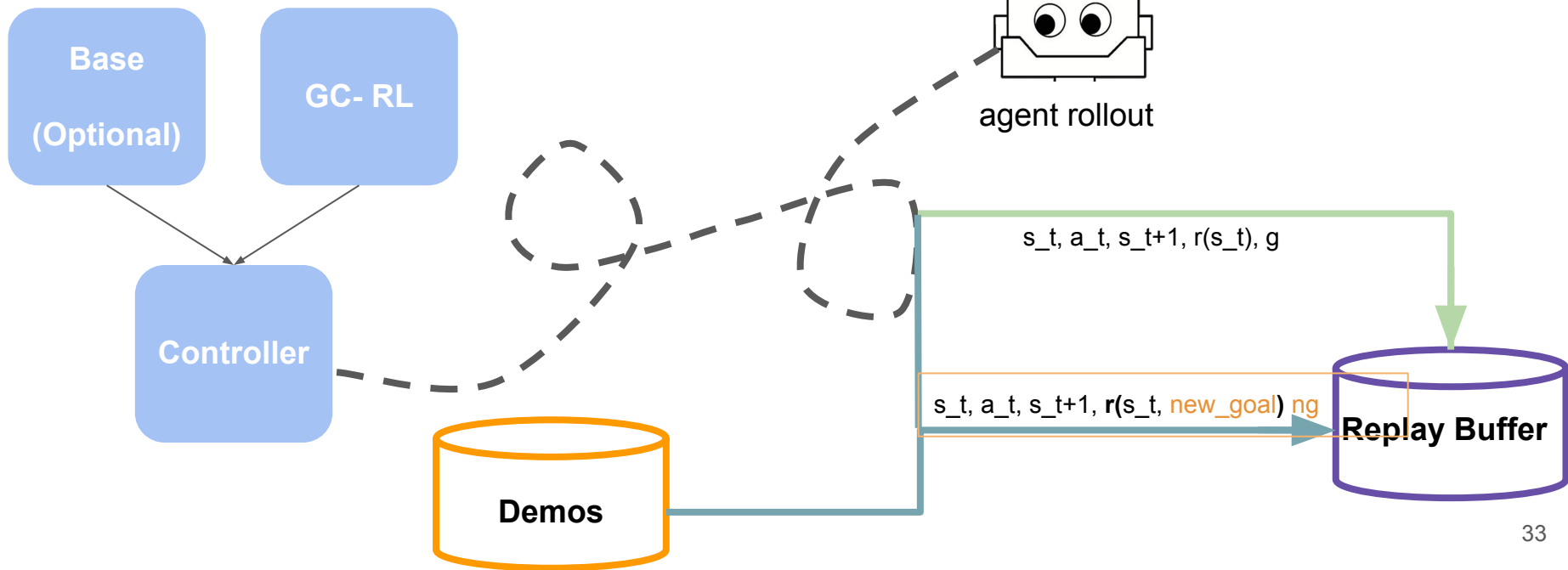
Hindsight Goal Selection for Demo-driven RL

(Key observations)

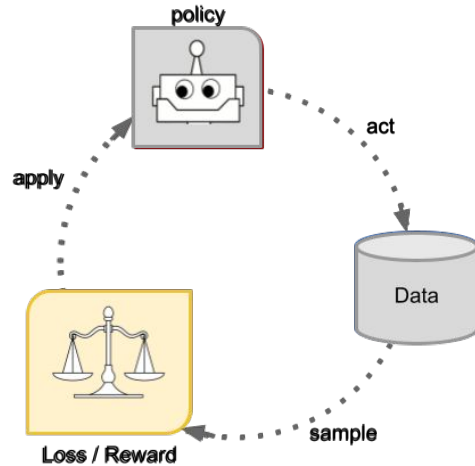


- A strategy for hindsight goal selection using demonstrations and distance-based gc reward
- A framework for task-constrained goal-conditioned RL
- That can successfully solve long-horizon dexterous manipulation tasks; and
- Works consistently and sample efficiently in challenging few-shot settings
- Robust towards using both engineered and learnt time-consistent representations

Quick Recap



Learning-to-learn Rewards

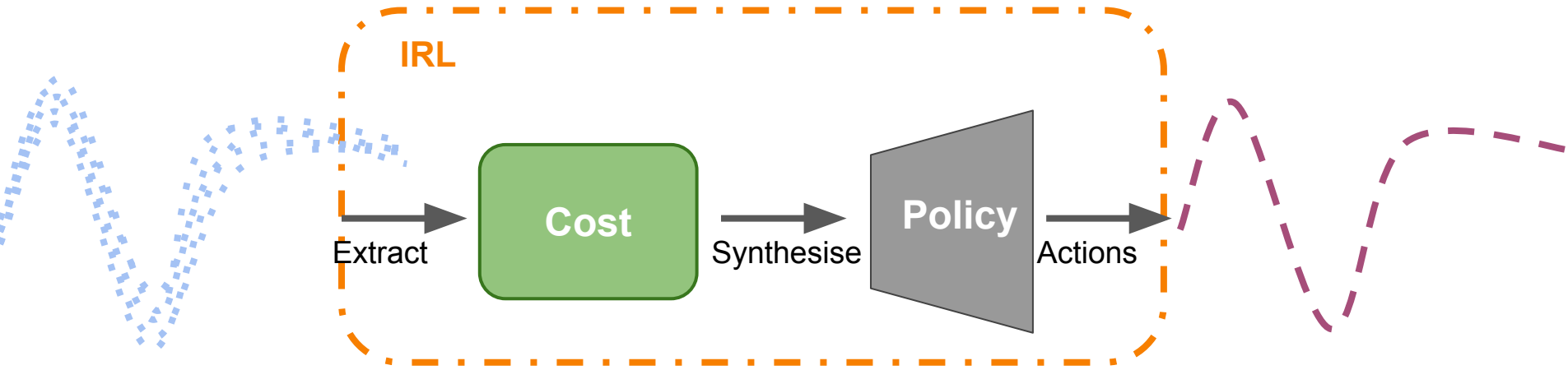


- we need more powerful reward functions;
- ideally, we should learn them;
- combining meta learning with reward learning can help;
- learn from visual demonstrations
- learn time-invariant rewards

Das, N., Behtle S., **Davchev T.**, Jayaraman D., Rai A., and Meier F., 2020. Model-based inverse reinforcement learning from visual demonstrations. CoRL 2020

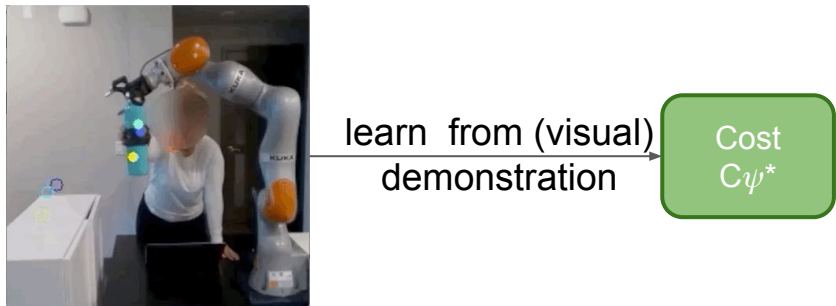
Davchev, T., Behtle, S., Ramamoorthy, S. and Meier, F., 2021. Learning Time-Invariant Reward Functions through Model-Based Inverse Reinforcement Learning. *arXiv preprint arXiv:2107.03186*.

Inverse Reinforcement Learning (IRL)



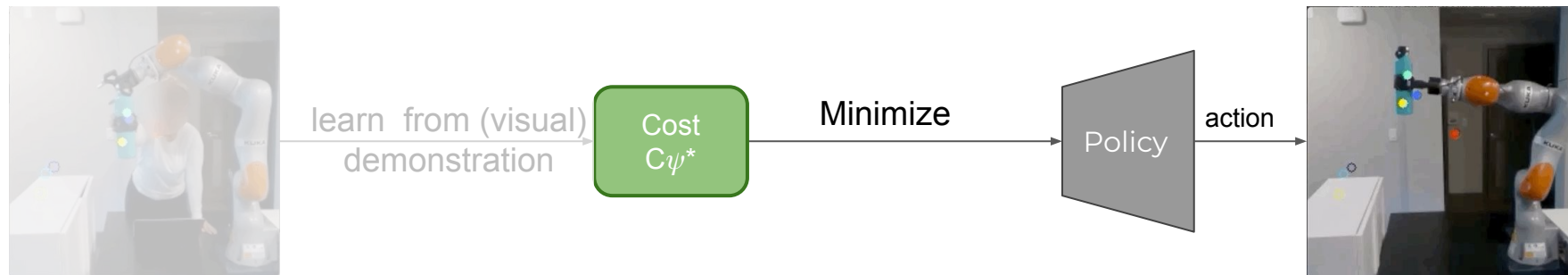
Goal: Learn Manipulation from Demonstrations via Model-based Inverse Reinforcement Learning

Step 1: learn cost function from (visual) demonstrations

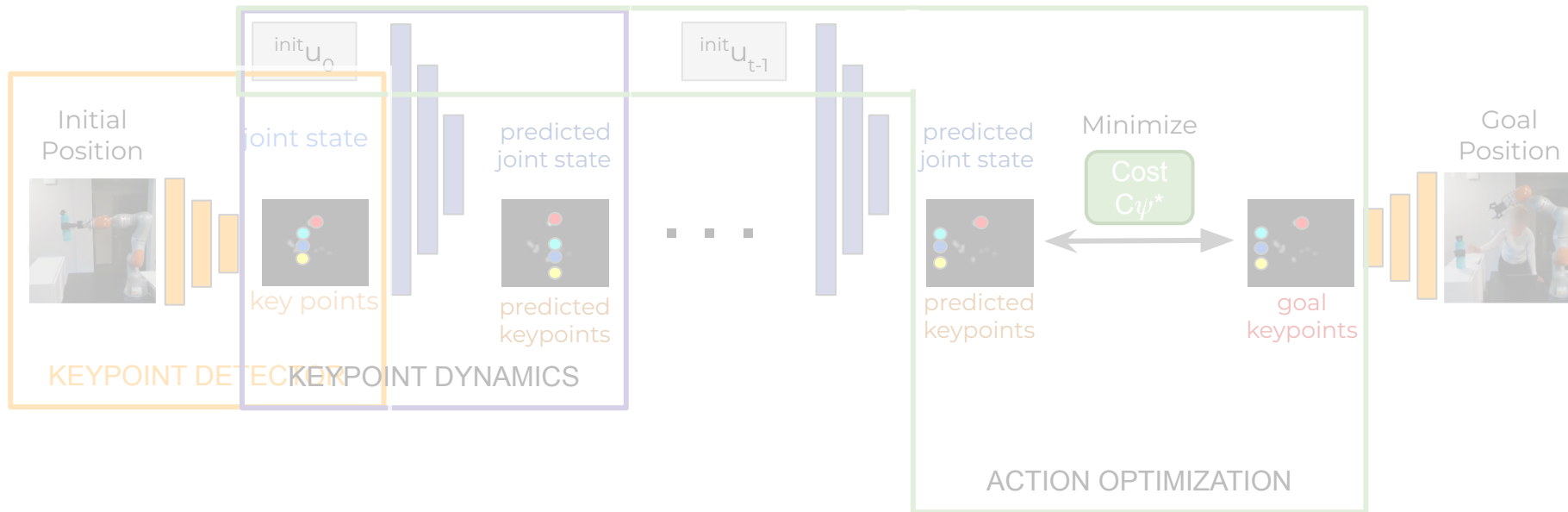


Goal: Learn Manipulation from Demonstrations via Model-based Inverse Reinforcement Learning

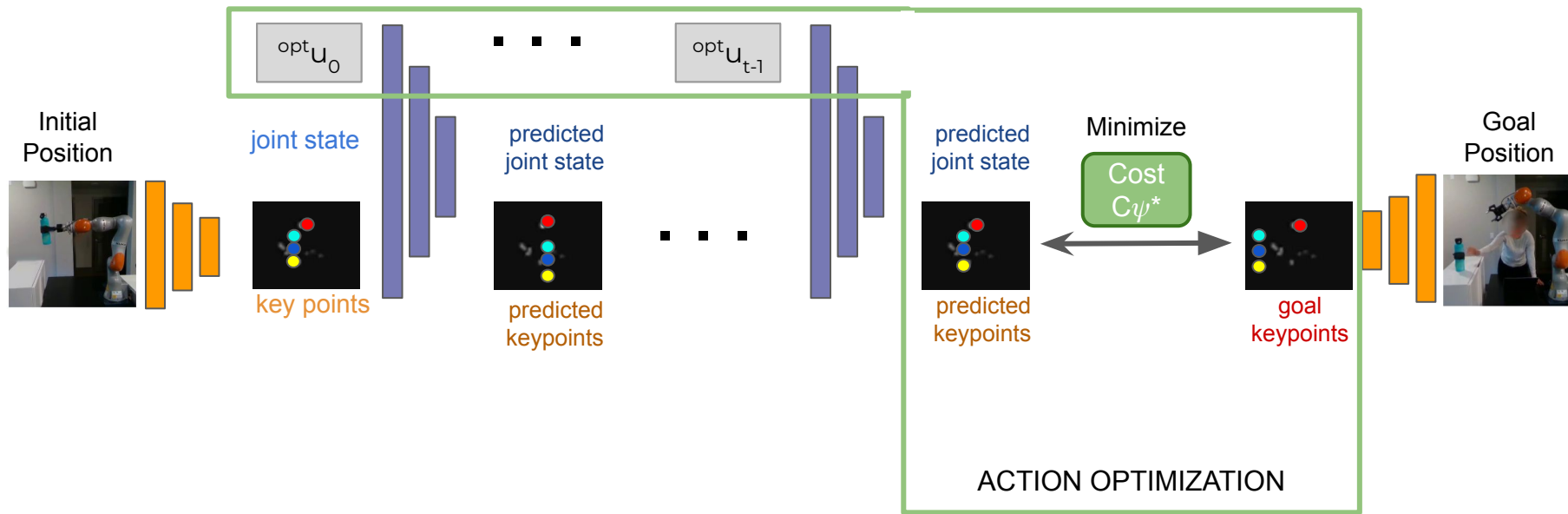
Step 2: reproduce behaviour by optimising actions wrt the learned cost



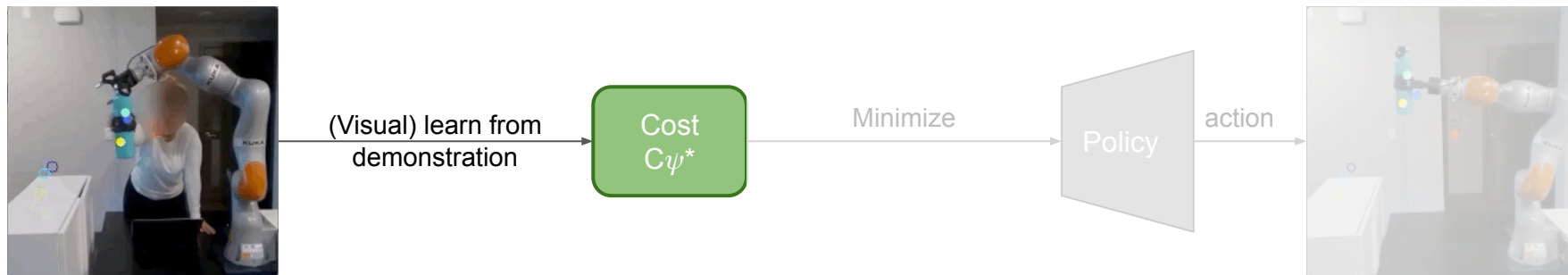
Optimising Actions via Visual MPC



Action Optimisation



IRL Overview: How to learn the cost function



IRL Overview: How to learn the cost function

BI-LEVEL OPTIMIZATION FRAMEWORK

Inner Loop: optimize actions u^{opt} by minimizing the cost $C\psi$ through a gradient update

$\nabla \psi^{\text{opt}u}$

Outer Loop: Update the cost parameters ψ , by minimizing the distance between the demonstration and the trajectory τ resulting from executing u^{opt}

IRL Overview: Cost function Representation

$$\text{Weighted Cost } C_{\psi}(\hat{\tau}, z_{\text{goal}}) = \sum_k \left[\psi_k^x \sum_t (\hat{z}_{t,k}^x - z_{\text{goal},k}^x)^2 + \psi_k^y \sum_t (\hat{z}_{t,k}^y - z_{\text{goal},k}^y)^2 \right]$$

$$\text{Time Dependent Weighted Cost } C_{\psi}(\hat{\tau}, z_{\text{goal}}) = \sum_k \sum_t \left[\psi_{t,k}^x (\hat{z}_{t,k}^x - z_{\text{goal},k}^x)^2 + \psi_{t,k}^y (\hat{z}_{t,k}^y - z_{\text{goal},k}^y)^2 \right]$$

$$\text{RBF Weighted Cost } C_{\psi}(\hat{\tau}, z_{\text{goal}}) = \sum_k \sum_t \sum_j \left[\psi_{j,k}^x(t) (\hat{z}_{t,k}^x - z_{\text{goal},k}^x)^2 + \psi_{j,k}^y(t) (\hat{z}_{t,k}^y - z_{\text{goal},k}^y)^2 \right]$$

$$\text{Default Cost } C_{\text{default}} = \sum_t^T (\hat{z}_t - z_{\text{goal}})^2$$

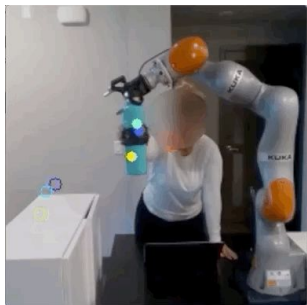
Results: Placing objects via Learned Cost

Human Demo

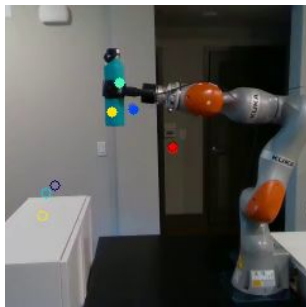
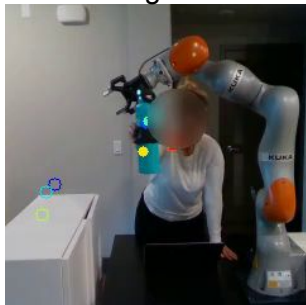


Results: Placing objects via Learned Cost

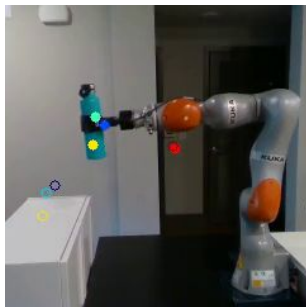
Human Demo



Human Demo -
Starting Point



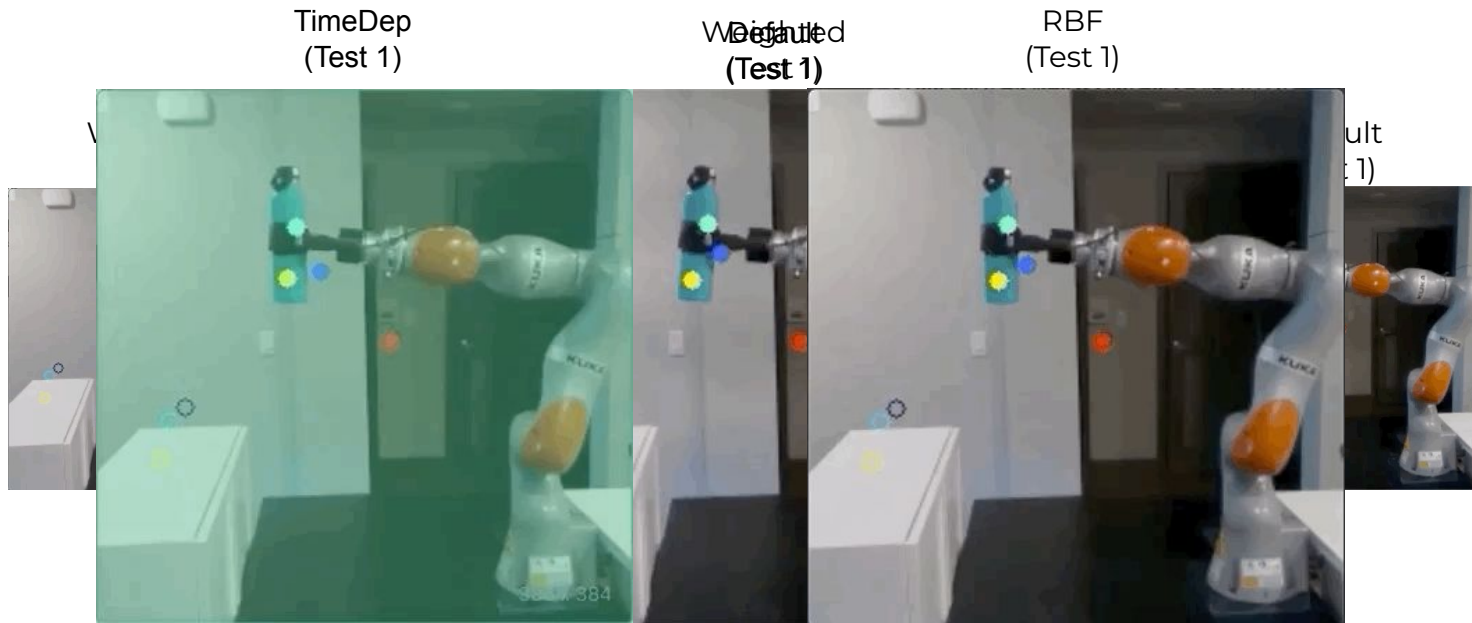
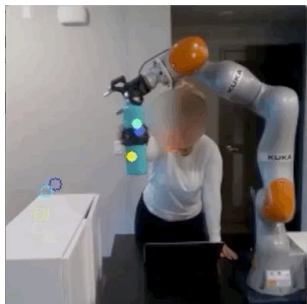
Test Case 1 - Starting Point 1



Test Case 2 - Starting Point 2

Shown in the Paper

Human Demo

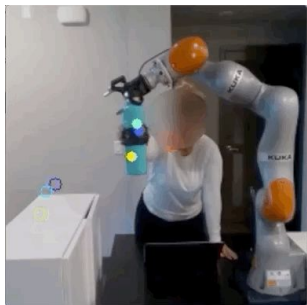


Start	Weighted Mean (Std)	TimeDep Mean (Std)	RBF Mean (Std)	Default Mean (Std)
1	40.99 (9.08)	6.12 (0.94)	4.26 (1.20)	26.96 (5.41)

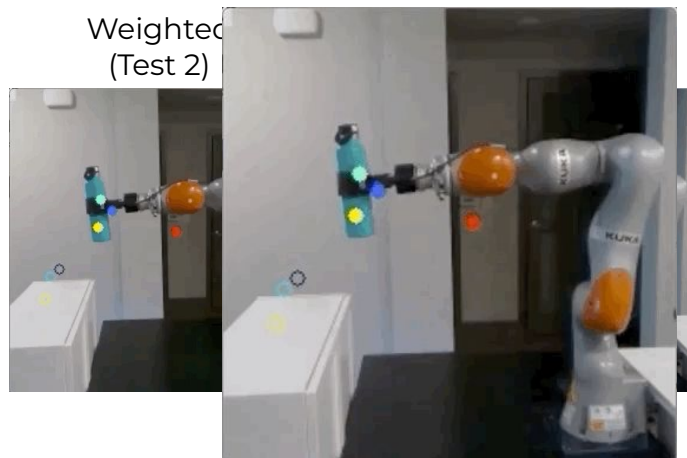
Not Shown in
the Paper

Start	Weighted Mean (Std)	TimeDep Mean (Std)	RBF Mean (Std)	Default Mean (Std)
2	3.61 (0.40)	3.53 (0.21)	4.40 (0.15)	15.76 (1.34)

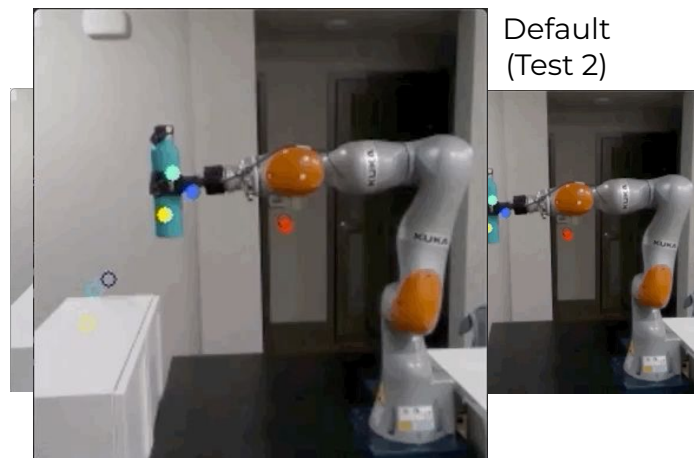
Human Demo



TimeDep
(Test 2)

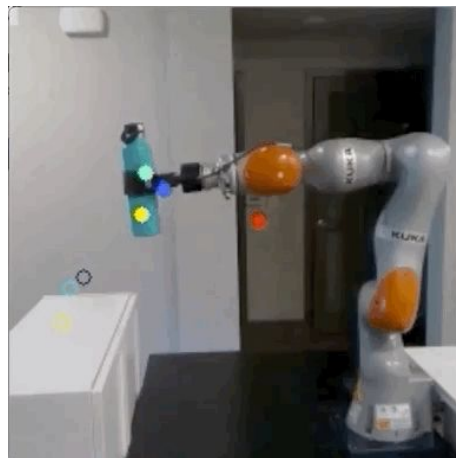


Default
(Test 2)



Model-based Inverse Reinforcement Learning Learning from Visual Demonstrations

(Key observations)

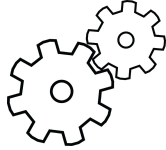


- gradient-based IRL framework that learns cost function from visual human demonstrations
- learn a compact keypoint-based image representation and train visual dynamics in that latent space
- learn different cost functions using our gradient-based IRL algorithm
- show that RBF weighted rewards perform best

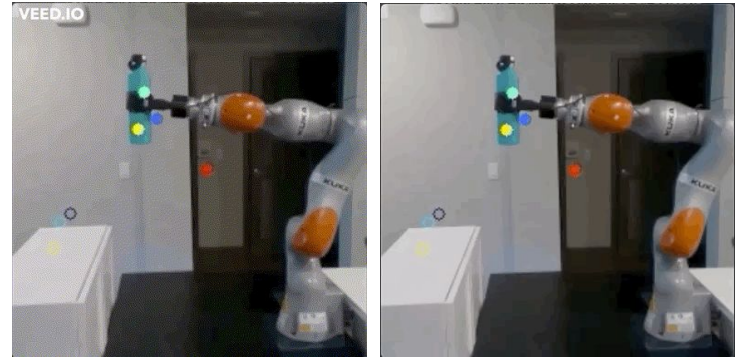
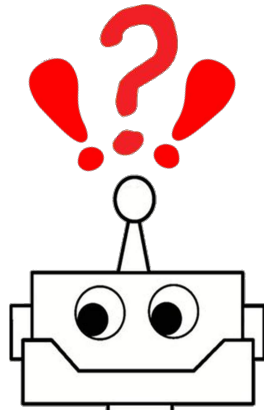
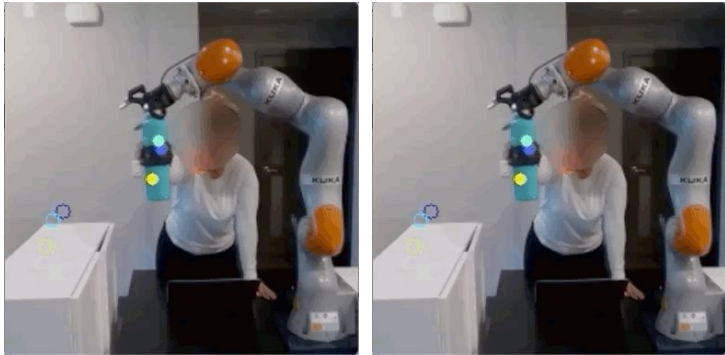
Motivation

Misaligned
Demonstrations

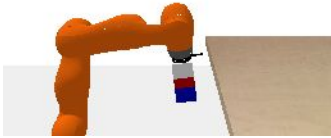
IRL



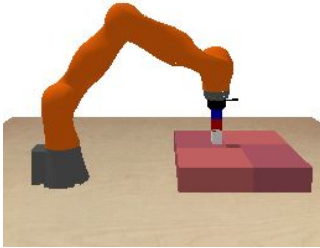
Vary
Execution Speed



Tasks



object placement



peg in hole

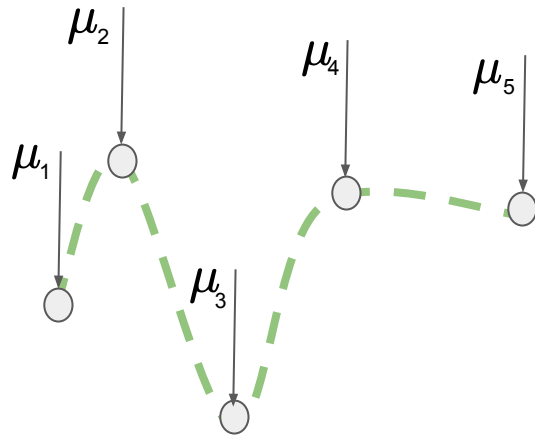
Practical Example



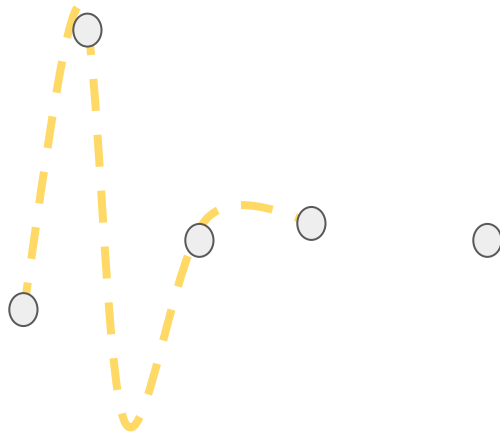
$$C_\phi(\hat{\tau}, g)$$

Basic structured Cost

$$C_\phi(\hat{\tau}, g) = \mathbb{E}_{\hat{\tau}} \left[\phi \cdot e^{b(x-\mu)^2} (\hat{\tau} - g)^2 \right]$$



However



Time-invariance approach

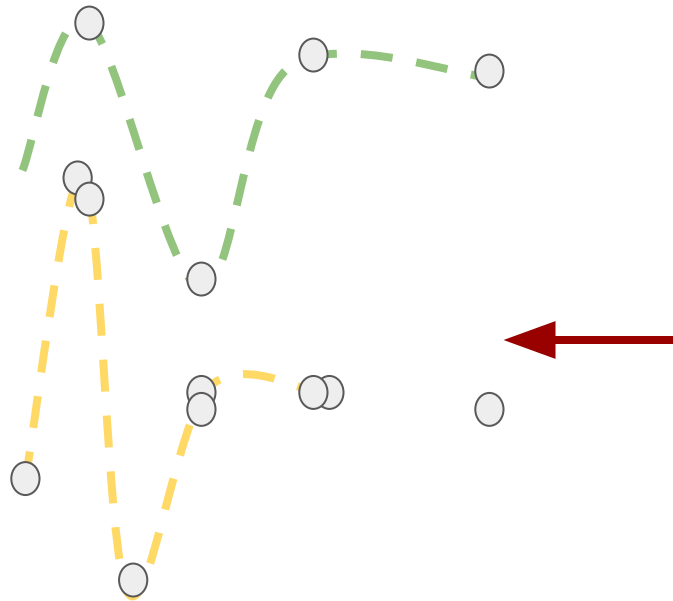


$$C_\phi(\lambda x, \hat{\tau}, g) = \mathbb{E}_{\hat{\tau}} \left[\phi \cdot e^{b(\lambda x - \mu)^2} (\hat{\tau} - g)^2 \right]$$

Structured Time-invariant cost

$$C_\phi(\lambda x, \hat{\tau}, g) = \mathbb{E}_\tau \left[\sigma_\phi(\lambda x, \hat{\tau}, g) \right]$$

Unstructured Time-invariant cost




$$\lambda = \frac{|\tau_{base}|}{|\hat{\tau}|}$$

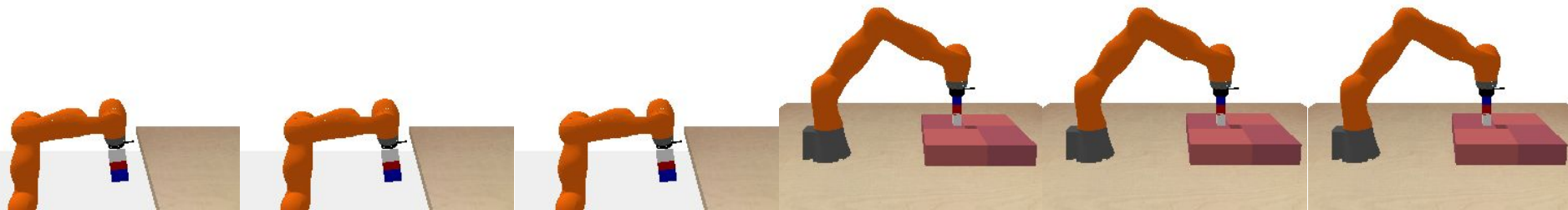
← avg. demo duration

← desired duration

Evaluation



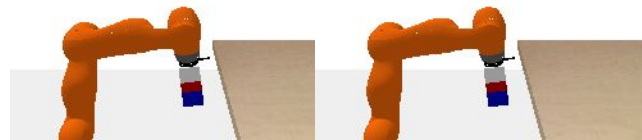
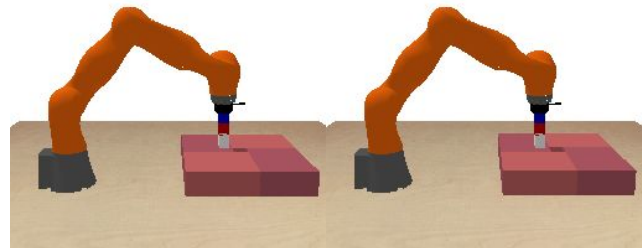
Env.	RBF [4] Mean (Std)	DTW-RBF Mean (Std)	λ -RBF Mean (Std)	MLP Mean (Std)	λ -MLP Mean (Std)
Placing	4.08cm (0.47)	5.88cm (0.25)	1.08cm (0.59)	4.10cm (0.02)	0.96cm (0.08)
Peg/Hole	22.22% (0.31)	25.1% (0.83)	59.33% (0.63)	32.45% (8.45)	70.22% (0.89)



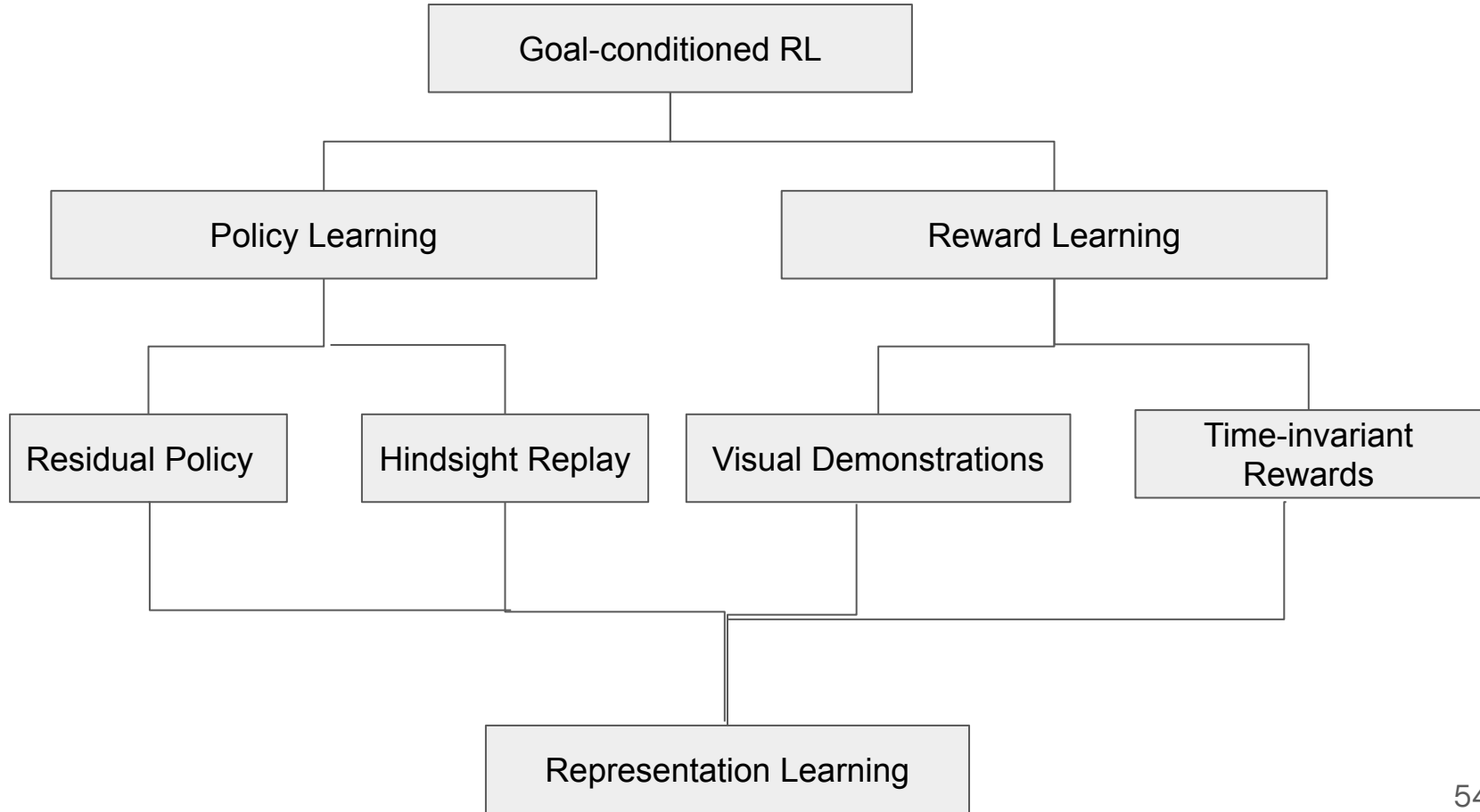
Learning Time-invariant Rewards with Model-based IRL

(Key observations)

- a temporal scaling mechanism for learning both time-invariant reward functions;
- learning from temporally misaligned demonstrations that can scale to different speeds;
- extensive comparison of the generalisation abilities of both structured and unstructured reward functions.



Summary



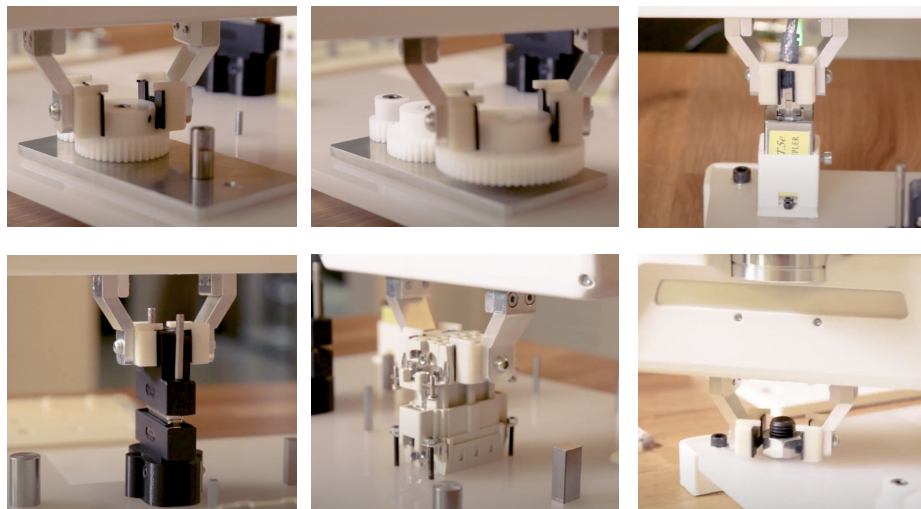
Potentially exciting directions...

LfD with informed representations



- fusing vision and force, e.g. using contrastive learning for imitation
- meta-learning for model-free LfD

Scale our policies to multi-task settings

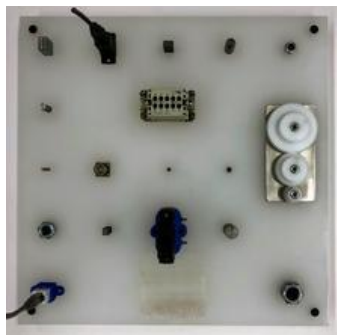


- utilise meta-learning for task-conditioned RL
- maintain multiple demo datasets and rewards

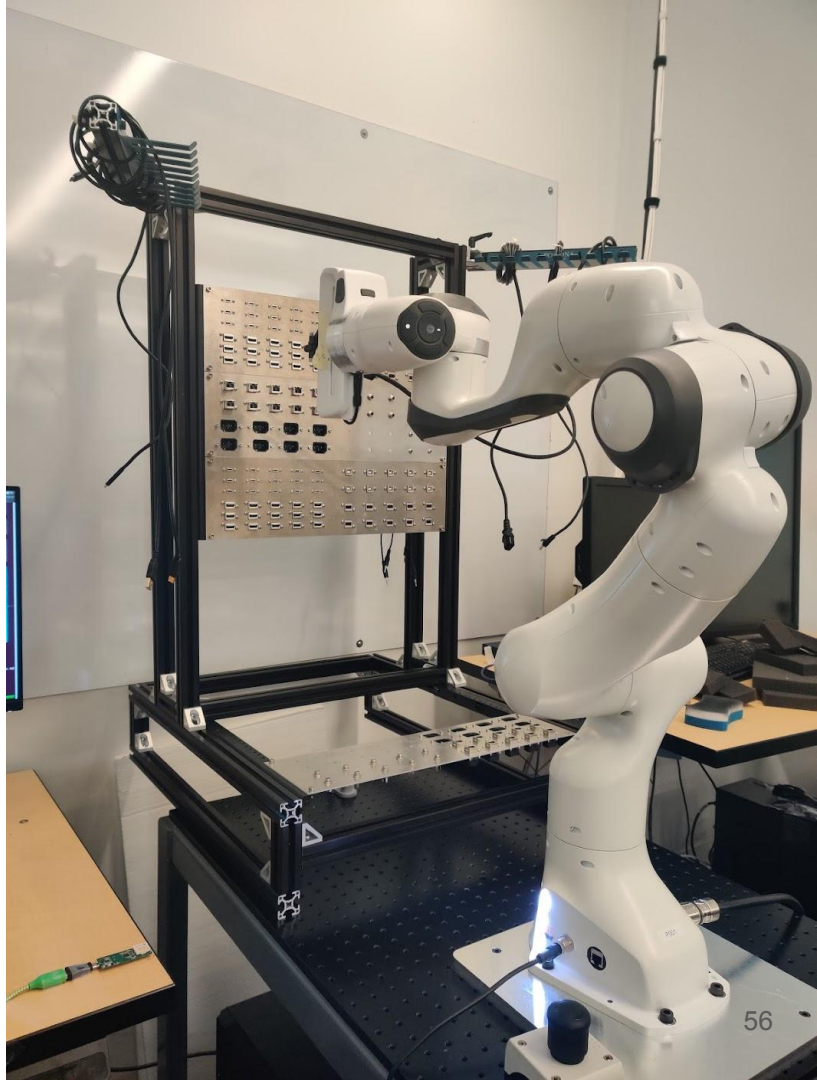
Future

- objective: enable robots to pick up skills similar to how humans do
- vision: enable sample-efficient lifelong learning of manipulation-based skills that can scale to complex sequential settings.
- next: the role of goal-conditioned RL for contact-rich manipulation;
 - scale to multi-task settings (i.e. lifelong learning);
 - utilise vision and touch for goal conditioned policy learning;
 - learn more informed goal-conditioned reward functions
- e.g., NIST challenge

Thank you!



email: t.davchev@gmail.com

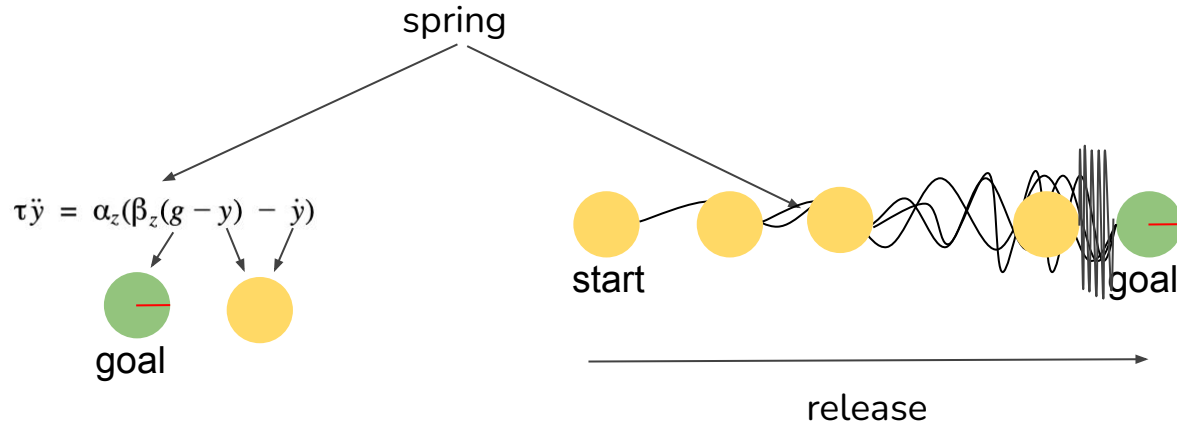


What are the Challenges

- adapt to model imperfections
- generalise to new settings
- adapt to world changes
- learn safe and efficient policies



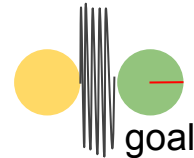
Dynamic Movement Primitives (recap)



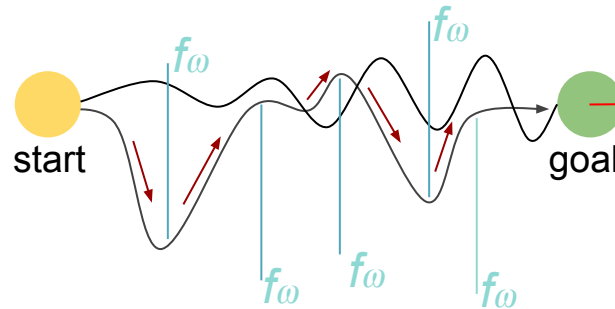


Dynamic Movement Primitives (recap)

$$\tau \ddot{y} = \alpha_z (\beta_z (g - y) - \dot{y})$$



$$\tau \ddot{y} = \alpha_z (\beta_z (g - y) - \dot{y}) + \mathbf{f}_\omega$$





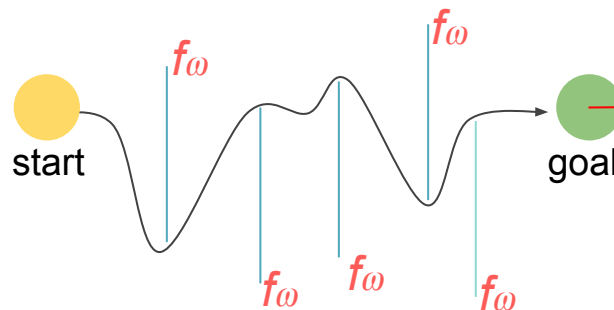
Dynamic Movement Primitives (recap)

$$\tau \ddot{y} = \alpha_z (\beta_z (g - y) - \dot{y})$$



$$\tau \ddot{y} = \alpha_z (\beta_z (g - y) - \dot{y}) + \mathbf{f}_\omega$$

Learn from Demonstration!

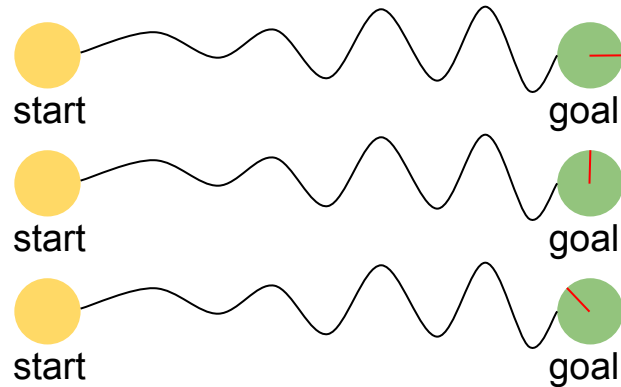




Reformulate in Quaternion (recap)

Orientation is not Commutative!

$$\tau^2 \dot{\omega} = a_{\omega} (\beta_{\omega} 2 \log(Q_g \circ Q^*) - \tau \omega) + f.$$



Residual Pose Corrections

Base policy Residual policy

Position

$$\pi_f = \pi_b + \pi_\theta$$

$\mathbf{a}_f^{\text{position}} = \mathbf{a}_b^{\text{position}} + \mathbf{a}_\Delta^{\text{position}}$

Orientation

$$\pi_f^{\text{orient}} = \pi_\theta^{\text{orient}} \circ \pi_b^{\text{orient}},$$

$$Q_f = Q_\Delta \circ Q_b.$$

BUT: quaternions are hypercomplex numbers!

Residual Orientation Corrections

$$\pi_f^{\text{orient}} = \pi_\theta^{\text{orient}} \circ \pi_b^{\text{orient}}$$

$$q_\Delta^w = \cos(\alpha/2.)$$

$$\vec{r}_\Delta = \frac{\omega^k}{\|\vec{\omega}\|}$$

Easier to learn for a
policy!

$$Q_\Delta = [\cos(\alpha/2.), \vec{r}_\Delta \sin(\alpha/2.)]$$

$$Q_f = Q_\Delta \circ Q_b$$

Residual Pose Corrections

Position

$$\pi_f = \pi_b + \pi_\theta$$

$$\mathbf{a}_f^{\text{position}} = \mathbf{a}_b^{\text{position}} + \mathbf{a}_\Delta^{\text{position}}$$

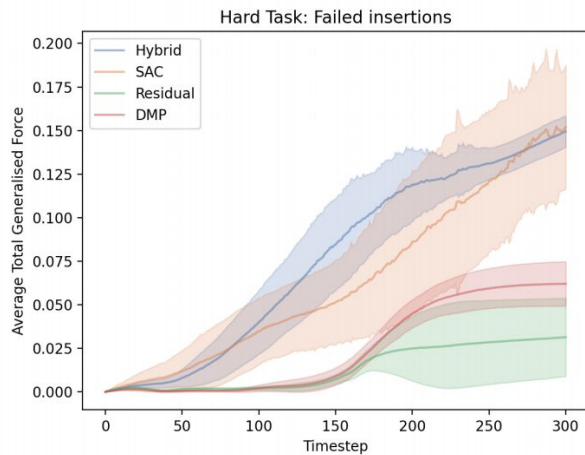
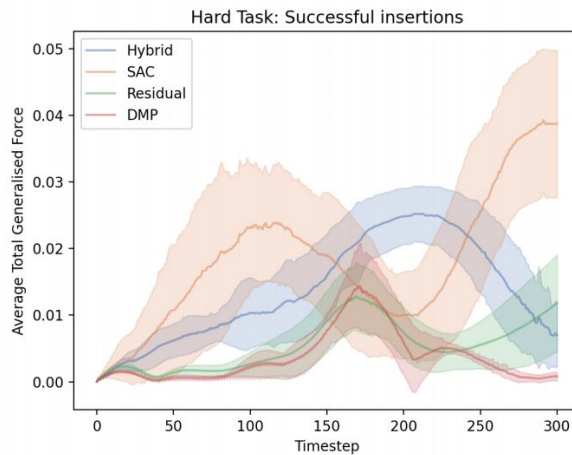
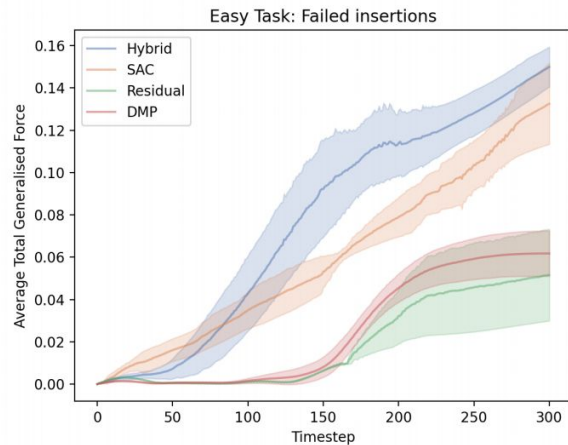
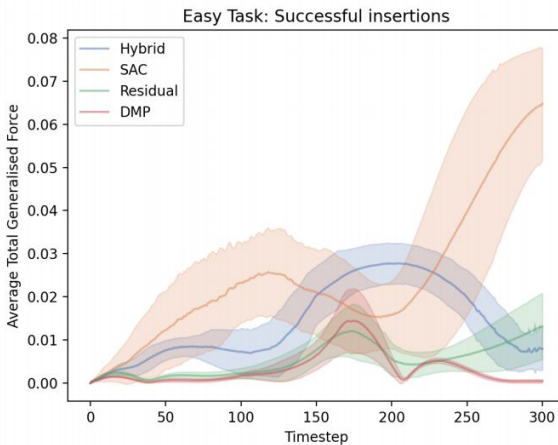
Orientation

$$\pi_f^{\text{orient}} = \pi_\theta^{\text{orient}} \circ \pi_b^{\text{orient}}$$

$$Q_f = Q_\Delta \circ Q_b$$

Gentle to the joints?

How much force on average would all joints experience?



Do we really need nonlinear policies?

Table 4.2: Residual adaptive policies in task space for DMPs (Exploration type colour from Fig. 5.1). Insertion accuracy table.

Type	Corrections	Adaptive Policy	Easy	Hard	Average	Eff.	Reward
A	translation	Random (Pottle, 1963)	25.8% \pm 4.3	9.0% \pm 1.8	17.4% \pm 3.1	n/a	n/a
A	translation	Linear (Kumar et al., 2016)	25.4% \pm 3.6	8.0% \pm 2.6	16.7% \pm 3.1	n/a	n/a
D	translation	SAC	94.8% \pm 1.3	55.0% \pm 2.7	74.9% \pm 2.0	700	$\mathbb{1}[L_2 \leq \kappa]$
D	translation	PPO	87.6% \pm 1.5	69.0% \pm 4.6	78.3% \pm 3.1	25.5K	$\mathbb{1}[L_2 \leq \kappa]$

RL in task space

Table 4.3: Comparing different ways of using nonlinear policies (Exploration type colour from Fig. 5.1). Insertion accuracy table.

Type	Corrections	Policy Type	Easy	Hard	Average	Eff.	Reward
A	None	DMP	24.0% \pm 2.5	8.0% \pm 1.4	16.0% \pm 2.0	n/a	n/a
D	translation	(pure) SAC (Haarnoja et al., 2018)	94.4% \pm 1.2	41.8% \pm 7.2	68.1% \pm 4.2	12.5K	$-(\alpha * L_1 + \frac{\beta}{L_2 - \epsilon})$
D	translation	(hybrid) SAC	57.2% \pm 2.5	45.6% \pm 2.7	46.4% \pm 2.6	8K	$\mathbb{1}[L_2 \leq \kappa]$
D	translation	(rLfD) SAC (ours)	94.8% \pm1.3	55.0% \pm2.7	74.9% \pm2.0	700	$\mathbb{1}[L_2 \leq \kappa]$

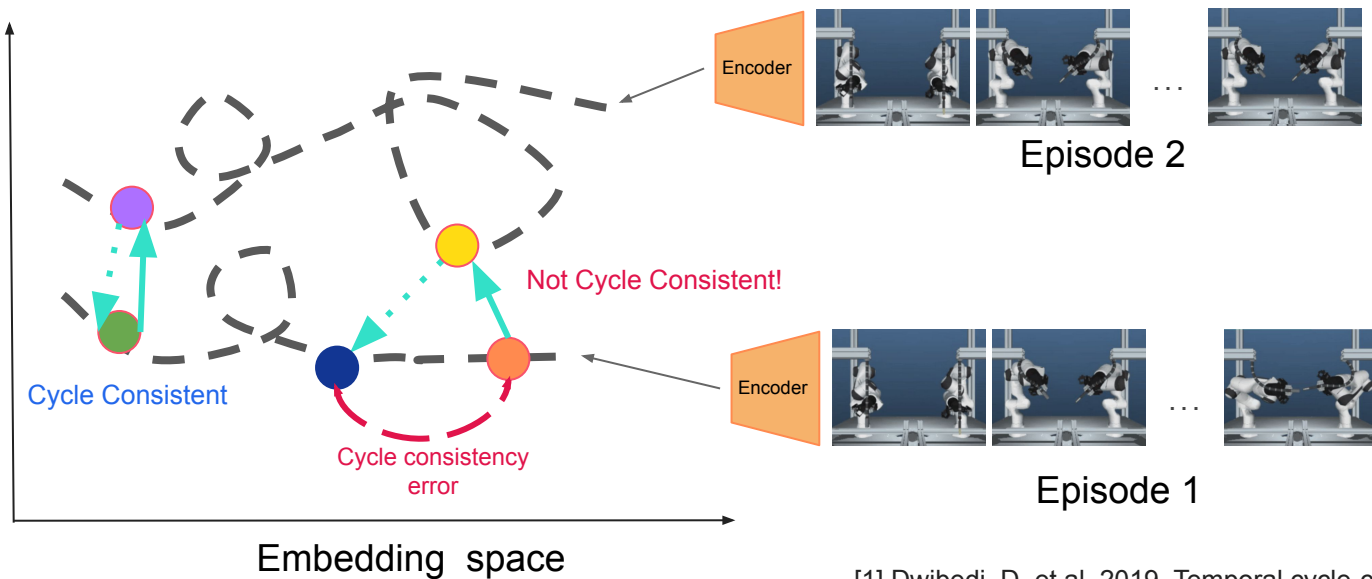
Time-consistent Representations

Engineering ψ :

- notion of progress - distance from target goal
- notion of contact - insertion was successful

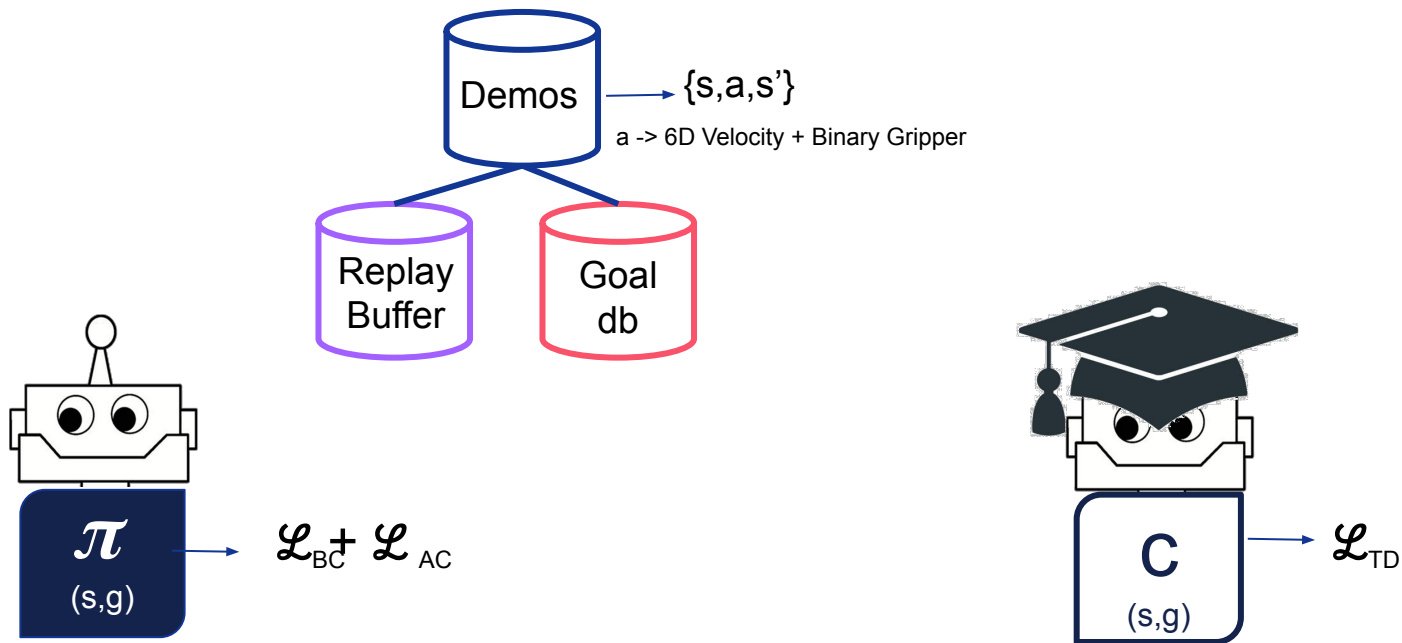
Learning ψ :

- Self-supervised learnt representation
- Directly used with provided demonstrations
- Refined over time
- Consistent + aligned notion of progress without labels
- Employ cycle consistency loss [1]



[1] Dwibedi, D. et al. 2019. Temporal cycle-consistency learning.

Goal-conditioned Distributional Policy Gradient from Demos



$\mathcal{L}_{BC} = \text{Regression} + \text{CE}$

$\mathcal{L}_{AC} = \text{DPG} + \text{SVG} + \text{L2}$

*Gumbel softmax trick for binary gripper

$\mathcal{L}_{TD} = \text{Distributional Critic} + \text{L2}$

- [1] Silver, D. et al. Deterministic policy gradient algorithms, ICML 2014
- [2] Heess, N. et al. Learning continuous control policies by stochastic value gradients, NeurIPS 2015
- [3] Jang, E. et al. Categorical reparameterization with gumbel-softmax, ICLR 2016
- [4] Bellemare, M.G. et al. A distributional perspective on reinforcement learning, ICML 2017
- [5] Vecerik, M. et al. A practical approach to insertion with variable socket position using deep reinforcement learning, ICRA 2019



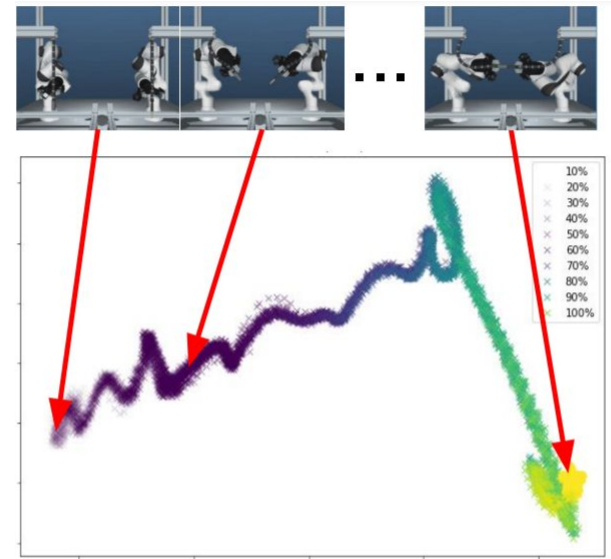
Distance-based Reward Specification

Utilise local smoothness of the representation space

Distance-based goal conditioned reward function

$$r(z, z_g) = \mathbb{1}[\|z - z_g\| < \epsilon]$$

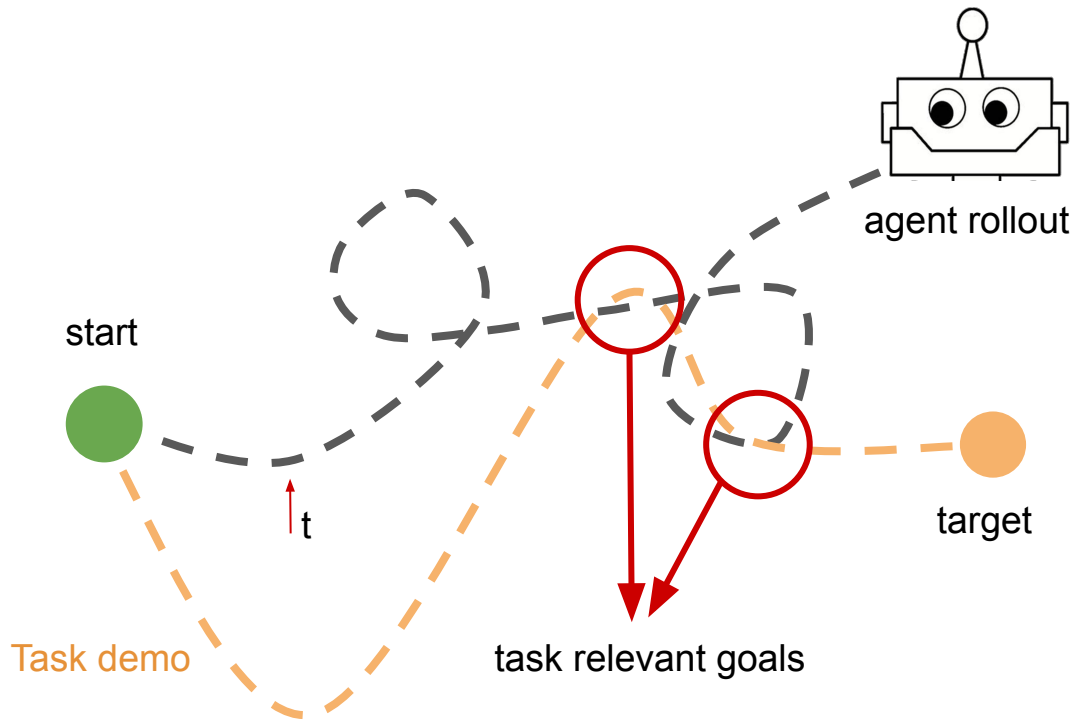
Encoded state Encoded goal Threshold obtained from demos



t-SNE visualisation of a temporally aligned representation space learnt with TCC

- Consistent notion of progress across 100 trajectories
- Can align != in length and motion trajectories that pass through similar stages

Constraining the self-supervision process



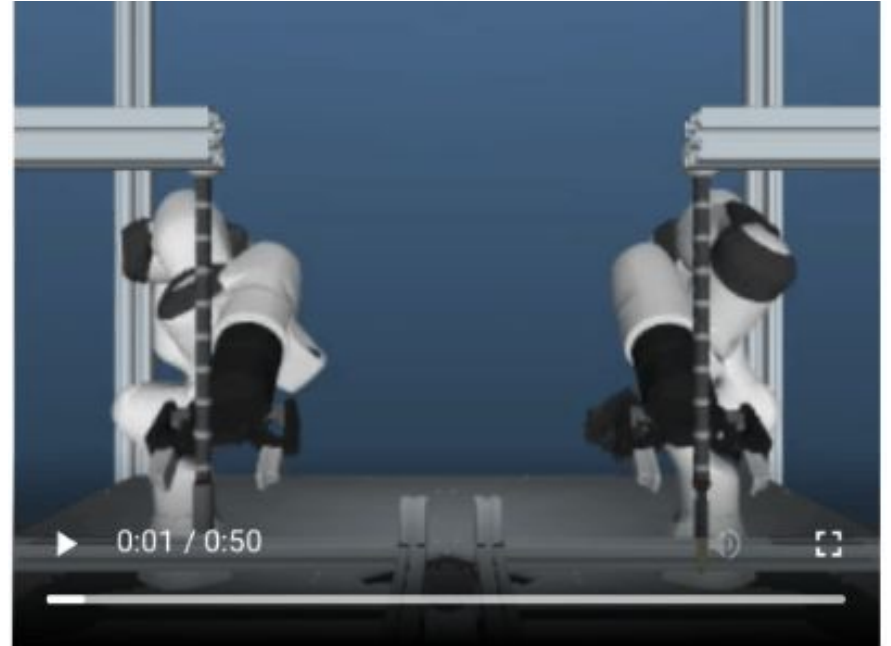
What benefits can we get from using task specific relabelling?

- Focus on targeted goals from demos
- Focus on feasible and relevant goals only
- Maintain benefits from HER

Experimental Evaluation

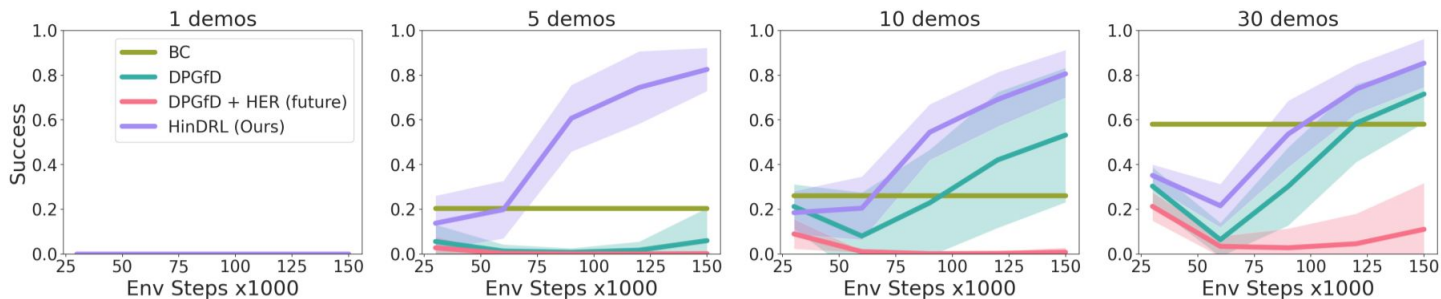
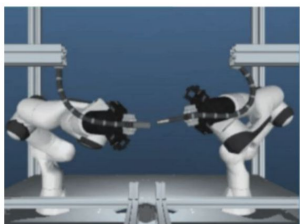


- Simple parameterised reach
- Reach + Grasp + Lift
- Reach + Grasp + Lift + Align
- Full Insertion

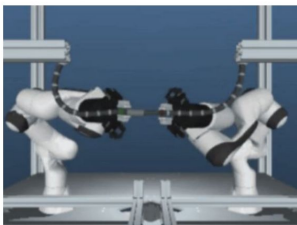


Few-shot tasks Performance: dependency over demonstration

Bring Near + Orient



Bimanual Insertion

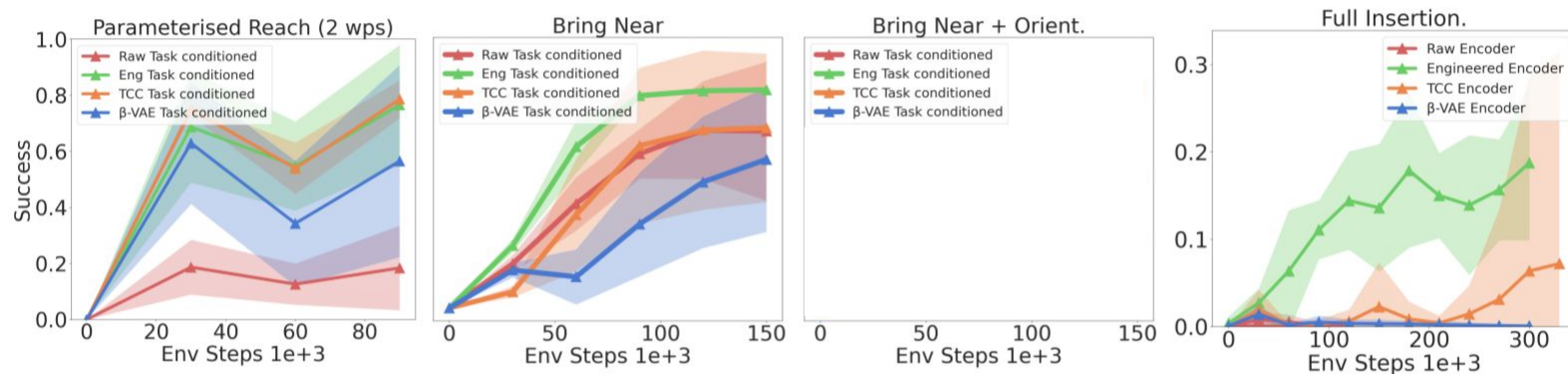


Environment	BC	DPGfD	HER (future)	HinDRL (Our)
	Mean (Std)	Mean (Std)	Mean (Std)	Mean (Std)
55 demos	16.06% (0.0)	4.28% (4.4)	18.39%	78.92% (10.3)
10 demos	2.94% (0.0)	0.0% (0.0)	0.0% (0.0)	32.05% (23.3)
5 demos	0.52% (0.0)	0.0% (0.0)	0.0% (0.0)	17.78% (18.9)
1 demos	0.13% (0.0)	0.0% (0.0)	0.0% (0.0)	12.58% (18.3)

HinDRL consistently works better than all baselines and is less dependent on the number of demonstrations used.

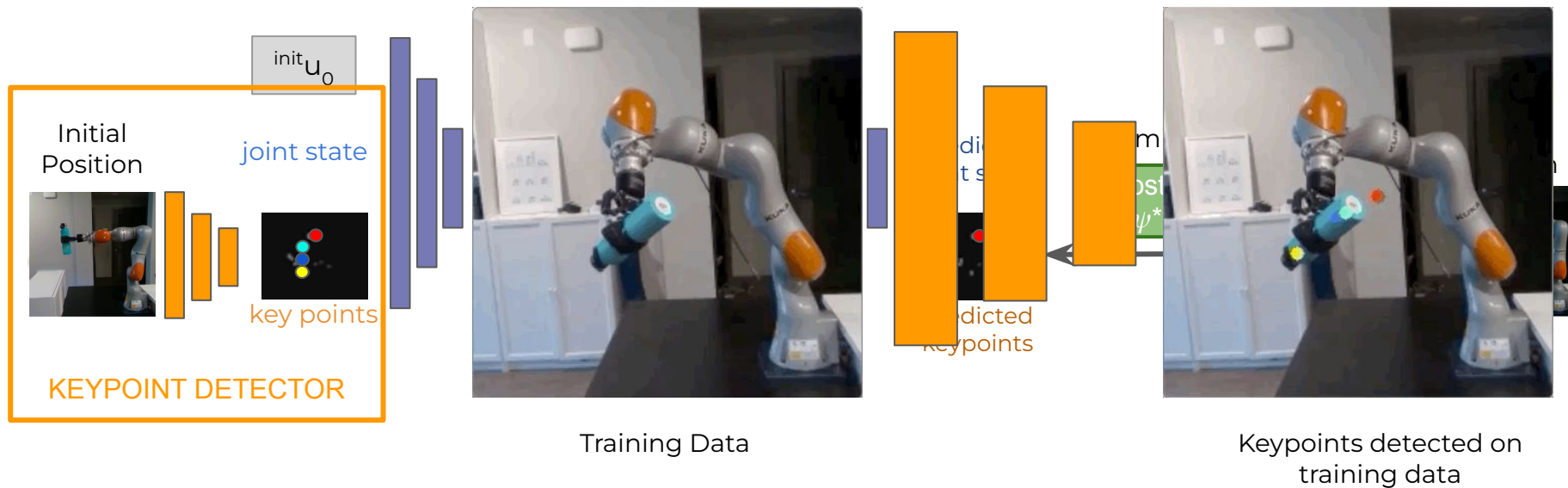
Robustness to the quality of the encoder

- State representations useful as input but also during relabelling for gc-reward.
- But access to near perfect representations is not always feasible
- Study sensitivity of the solution to the quality of the encoder.

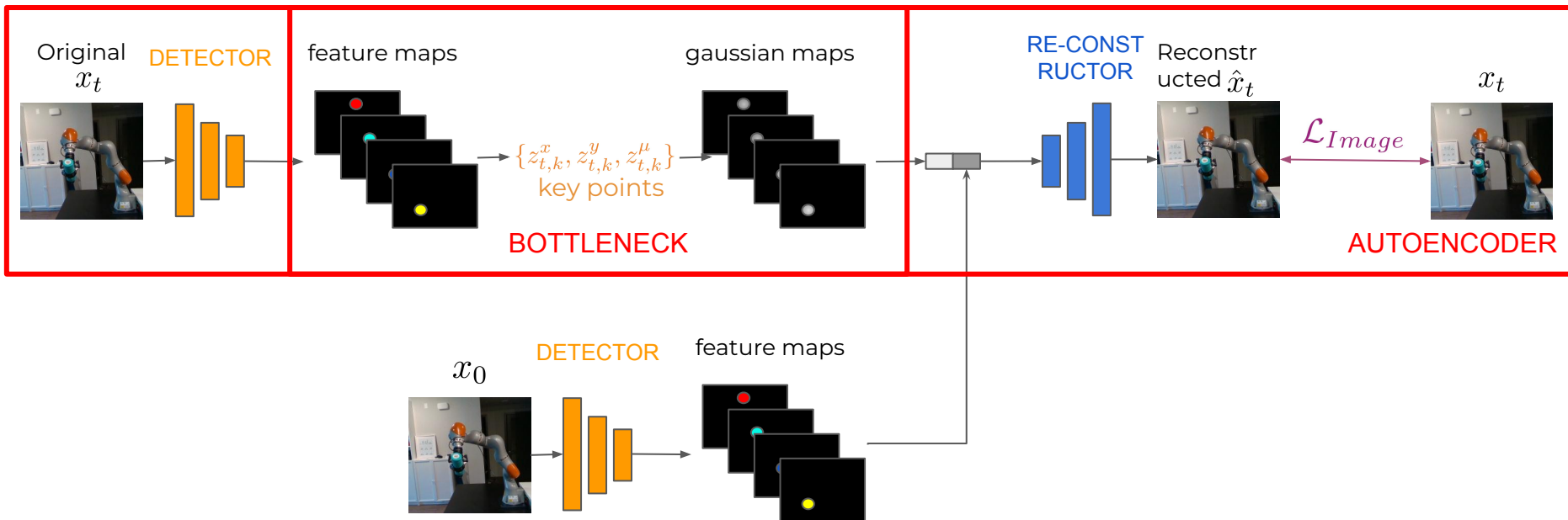


Notion of progress is central to the success of HinDRL!

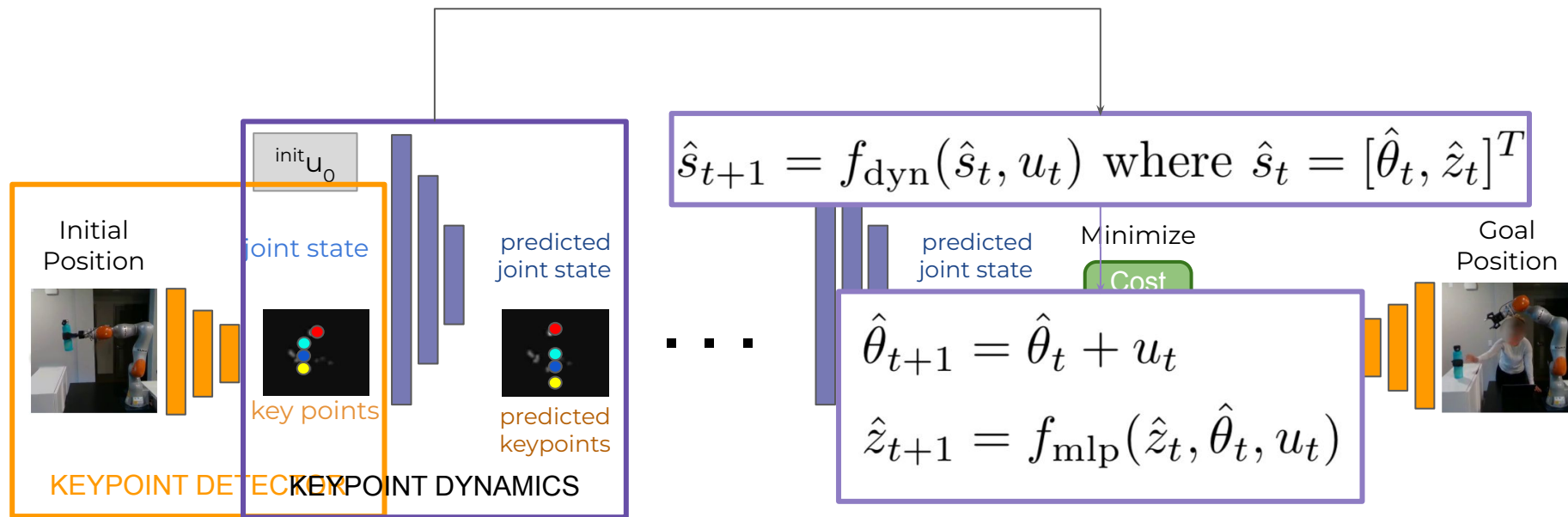
Training the keypoint detector



Training the keypoint detector - Framework

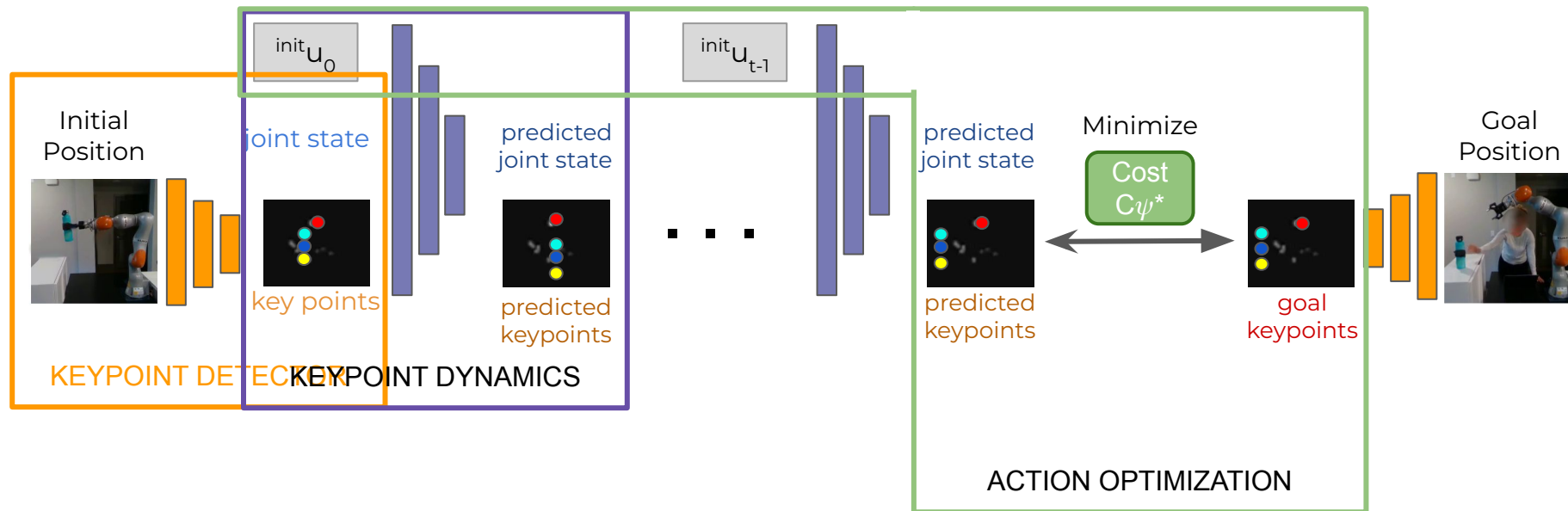


Training the Dynamics Model



f_{mlp} is an MLP with 2 linear layers with ReLU activations, containing 100 and 25 neurons respectively

Optimizing Actions via Visual MPC



Evaluation



Tasks varied by:

- the execution speed
- distance to goal
- Total of 30 different tasks

Meta-training:

- three different training routines
- report distance from desired / target speed

Fixed speed and mixed goals (standard)

Mixed speed but fixed goals

Mixed speed and mixed goals

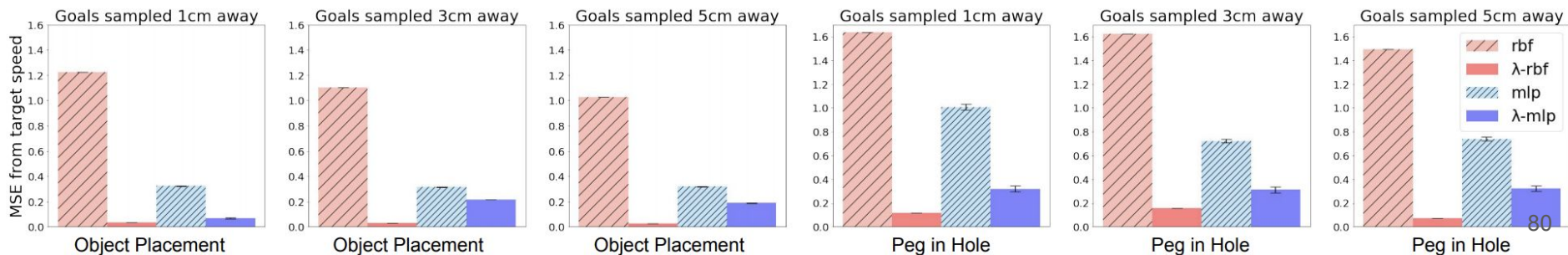
Evaluation (meta test)

Overall Performance:

- Trained on misaligned demonstrations
 - (3 and 5 second executions)
 - demo goals sampled up to 1cm away
- vary tasks in terms of speed / goal location
 - 2-6 second long executions
 - goals sampled up to 5cm away
 - measure task success
 - measure distance to target speed

Env.	RBF [4] Mean (Std)	DTW-RBF Mean (Std)	λ -RBF Mean (Std)	MLP Mean (Std)	λ -MLP Mean (Std)
Placing	4.08cm (0.47)	5.88cm (0.25)	1.08cm (0.59)	4.10cm (0.02)	0.96cm (0.08)
Peg/Hole	22.22% (0.31)	25.1% (0.83)	59.33% (0.63)	32.45% (8.45)	70.22% (0.89)

TABLE II: Records performance in terms of mean squared distance in the placing environment and in terms of successful insertions for the peg in hole (PiH) environment. Test time table.



Evaluation (meta test)

Generalizing to different goals:

- assume fixed center of the goal distribution
- training data sampled within 1cm away from center
- evaluated on new goals sampled from up to 5 cm away
- goals sampled 1-3 or 3-5 cm away are out of the training distribution
- within 1cm are in-training

Task	RBF [4] Mean (Std)	DTW-RBF Mean (Std)	λ -RBF Mean (Std)	MLP Mean (Std)	λ -MLP Mean (Std)
1cm	26.0% (0.0)	44.0% (0.0)	86.0% (0.0)	28.0 (12.73)	78.0% (9.43)
3cm	10.67% (0.94)	15.3% (2.49)	44.0% (0.0)	30.67% (4.46)	74.0% (3.77)
5cm	39.0% (0.0)	16.0% (0.0)	48.0% (1.89)	38.67% (8.15)	58.67% (0.94)
Avg.	22.22% (0.31)	25.11% (0.83)	59.33% (0.63)	32.45% (8.45)	70.22% (0.89)

Task	RBF [4] Mean (Std)	DTW-RBF Mean (Std)	λ -RBF Mean (Std)	MLP Mean (Std)	λ -MLP Mean (Std)
1cm	3.49cm (2.87)	5.58cm (3.45)	0.35cm (0.31)	4.08cm (2.30)	0.87cm (0.30)
3cm	4.13cm (2.73)	5.85cm (3.32)	1.09cm (0.20)	4.10cm (2.24)	0.94cm (0.55)
5cm	4.63cm (2.63)	6.20cm (3.16)	1.79cm (0.16)	4.13cm (2.20)	1.07cm (0.71)
Avg.	4.08cm (0.47)	5.88cm (0.25)	1.08cm (0.59)	4.1cm (0.02)	0.96cm (0.08)