A Dual Representation Framework for Robot Learning with Human Guidance Appendix

1 Appendix: Tasks

We use five continuous control tasks to evaluate proposed algorithms and baselines: `Lunar-Lander`, `Reaching-Sim`, `Reaching-Real`, `Placing-Sim`, and `Placing-Real`. We use Robosuite [1] as the simulation platform for the reaching and placing. Below, we provide definition, visualization, and the high-level symbolic scene graph representation for each task.

1.1 Lunar-Lander-Continuous-v2

We use OpenAI Gym's Lunar-Lander-Continuous-v2 [2] as shown in Fig. 1. This is a challenging continuous control task with a complicated reward function. The goal is to control the lander to land safely in the middle of two flags without crashing. For the low-level representation, we follow the default setting [2]. The Markov decision process (MDP) of this task is as follows:

1. $\mathcal{S}$ is 8-dimensional: the coordinates of the lander in x and y directions, its linear velocities in x and y directions, its angle, its angular velocity, and two booleans that represent whether each leg is in contact with the ground or not.
2. $\mathcal{A}$ is 2-dimensional: the first dimension of an action determines the throttle of the main engine (-1 to 0: engine off; 0 to +1: throttle from 50% to 100% power). The second dimension specifies the throttle of the lateral boosters (-1.0 to -0.5: fire left engine; +0.5 to +1.0: fire right engine; -0.5 to +0.5: engines off).
3. $\mathcal{R}$: Reward for moving from the top of the screen to the landing pad and zero speed is about 100 to 140 points. If the lander crashes or comes to rest, it receives additional -100 or +100 points. Each leg's ground contact is +10. Firing the main engine is -0.3 points each frame.
4. $\gamma = 0.99$.

Recall our high-level symbolic scene graph $\mathcal{G}$ is represented as a binary vector of $dim(\mathcal{G})$, where each dimension represents a unary state of an object or a pairwise semantic relation between the objects and the robot. For `Lunar-Lander`, details of these dimensions are explained in Table 1. At a given time t, these binary values can be calculated from the low-level state information at time t, resulting in an abstract state g_t.

1.2 Reaching-Sim and Reaching-Real

In the reaching tasks, as shown in Fig. 2a, the goal is to move the robot's gripper to a target position marked on the table. The MDP of this task is as follows:

1. $\mathcal{S}$ is 2-dimensional: The robot end effector's position in the xy-plane. The height of the end effector above the table is fixed.
2. s_0: The robot's end effector position is initialized randomly in the area surrounded by the red, green, and blue lines in Fig. 2a.
3. $\mathcal{A}$ is 2-dimensional: The end effector $\Delta x, \Delta y$ movements in the xy-plane. In `Reaching-Sim`, these are in the range of $[-0.1, 0.1]cm$. In `Reaching-Real`, the end effector $\Delta x, \Delta y$ inputs to the impedance controller are in the range of $[-5, 5]cm$. The decision frequency is $5Hz$. In Reaching-Sim, the joint torque commands are computed using Robosuite's operational space controller [1]. For Reaching-Real, we implement an impedance controller. .

Dimension	State	Meaning
0	`left(lander,flag1)`	whether the lander is on the left side of flag1
1	`left(lander,flag2)`	whether the lander is on the left side of flag2
2	`above(lander,flags)`	whether the lander is above the flags
3	`left(lander,pad)`	whether the lander is on the left side of the landing pad
4	`above(lander,pad)`	whether the lander is above the landing pad
5	`upright(lander)`	whether the angle of the lander is in $[-0.5, 0.5]$ radians
6	`on(mainEngine)`	whether the lander's main engine is on
7	`on(leftEngine)`	whether the lander's left engine is on
8	`on(rightEngine)`	whether the lander's right engine is on
9	`contact(leftLeg,ground)`	whether the lander's left leg is in contact with the ground
10	`contact(rightLeg,ground)`	whether the lander's right leg is in contact with the ground
11	`fast(lander)`	whether the lander's velocity is greater than 1.0

Table 1: The symbolic scene graph of `Lunar-Lander` is represented as a 11-dimensional binary vector.

Dimension	State	Meaning
0	`left(gripper,redLine)`	whether the gripper is on the left side of the red line
1	`left(gripper,greenLine)`	whether the gripper is on the left side of the green line
2	`inFront(gripper,blueLine)`	whether the gripper is in front of the blue line
3	`inFront(gripper,blackLine)`	whether the gripper is in front of the black line

Table 2: The symbolic scene graph of `Reaching-Sim` and `Reaching-Real` is represented as a 4-dimensional binary vector.

4. $\mathcal{P}$: A hard-coded rectangular boundary is defined to restrict the robot's workspace to be within the table boundary. An action is ignored if it tries to take the robot gripper outside this boundary.
5. $\mathcal{R}$: When the robot's end effector moves inside a 10cm by 25cm target area located at the center of the table, a completion reward of +1 is given. The reward is 0 in all other states.
6. $\gamma = 0.99$.

The high-level representation is straightforward in these tasks, details are explained in Table 2.

1.3 Placing-Sim and Placing-Real

The goal for these tasks is to drop a ball through a hoop (Fig. 2b). We assume that the robot starts with the ball in its gripper, and the position of the hoop is fixed and known. The MDP of this task is as follows:

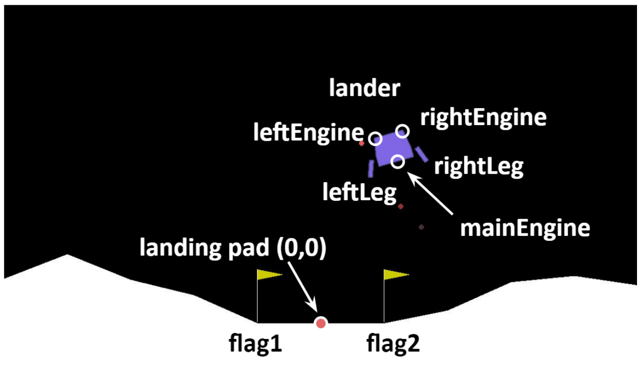
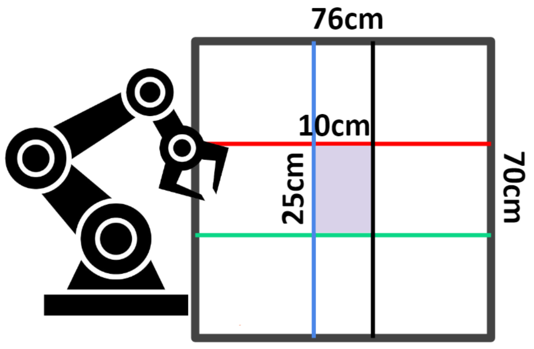
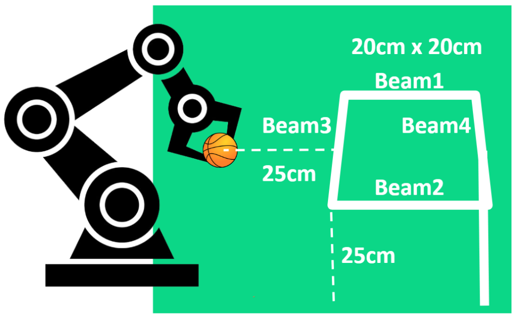


Figure 1: Task `Lunar-Lander`.



(a)



(b)

Figure 2: (a) Tasks `Reaching-Sim` and `Reaching-Real`. (b) Task `Placing-Sim` and `Placing-Real`.

Dimension	State	Meaning
0	`left(gripper,beam1)`	whether the gripper is on the left side of beam1
1	`left(gripper,beam2)`	whether the gripper is on the left side of beam2
2	`inFront(gripper,beam3)`	whether the gripper is in front of beam3
3	`inFront(gripper,beam4)`	whether the gripper is in front of beam4
4	`above(gripper,hoop)`	whether the gripper is above the hoop
5	`inside(gripper,ball)`	whether the ball is grasped by the gripper

Table 3: The symbolic scene graph of `Placing-Sim` and `Placing-Real` is represented as a 6-dimensional binary vector.

1. $\mathcal{S}$ is 4-dimensional: The robot end effector's position in 3D Cartesian space, and the gripper state.
2. s_0: The robot's end effector position is randomly initialized in the yz-plane. The x distance is $25cm$ from the center of the hoop.
3. $\mathcal{A}$ is 4-dimensional: The end effector $\Delta x, \Delta y, \Delta z$ movements in the 3D space, and the gripper command in the range of $[-1, 1]$. In Placing-Sim, $\Delta x, \Delta y, \Delta z$ are in the range of $[-0.1, 0.1]cm$. In Placing-Real, $\Delta x, \Delta y$ are in the range of $[-4, 4]cm$. Δz is in the range of $[-8, 8]cm$. The decision frequency is $5Hz$. The joint torque commands are computed in the same way as the reaching tasks.
4. $\mathcal{P}$: A rectangular boundary limits the robot's workspace in each of the 3 Cartesian dimensions to avoid joint limits being reached. An additional boundary is defined around the hoop to generate a no-entry zone, preventing collision between the robot and the hoop object. An action is ignored if it tries to take the robot gripper outside the boundary or into the no-entry zone. If an unexpected collision occurs, the episode is terminated.
5. $\mathcal{R}$: When the robot releases the ball into the hoop from above the hoop, a completion reward of +1 is given. If the robot drops the ball outside of the hoop, a negative reward of -1 is given. The reward is 0 in all other states.
6. $\gamma = 0.99$.

The high-level representation is also straightforward in these tasks, details are explained in Table 3.

2 Appendix: Evaluative Feedback

Synthetic humans for simulation experiments Using synthetic humans as the first step of our experiment allows a comprehensive evaluation of the algorithms. In our case, a fully trained SAC agent (an oracle) is used as the synthetic human. For a given state s, the learning agent chooses an action a based on its policy, and the oracle chooses an action a^* based on its near-optimal policy. We use the oracle SAC to compute the Q values for these actions: $Q(s, a)$ and $Q(s, a^*)$. To simulate

Algorithm	Parameter Name	Value
All	Learning rate	7.5e-4
All	Replay buffer size	1e6
All	Batch size	512
All	Gradient steps	8
All	`Lunar-Lander` hidden layer sizes	[400, 300]
All	`Reaching` hidden layer sizes	[64, 64]
All	`Placing` hidden layer sizes	[128, 128]
All	`Lunar-Lander` training steps	150,000
All	`Reaching-Sim, Placing-Sim` training steps	100,000
All	`Reaching-Real` training steps	1,250
All	`Placing-Real` training steps	2,500
EF, DREF	Threshold for positive feedback (α in Appendix Eq.1)	0.999 + 1e-8$\times T$
EF, DREF	Human feedback loss weight (λ in Eq.3)	9
DREF	`Lunar-Lander` rank threshold (k in Algorithm 1)	60
DREF	`Reaching` rank threshold (k in Algorithm 1)	2
DREF	`Placing` rank threshold (k in Algorithm 1)	4

Table 4: The list of hyperparameters for learning from evaluative feedback algorithms and experiments.

human feedback, our intuition is that if the learning agent chooses an action that has a Q-value close enough to $Q(s, a^*)$, it is a good action and the agent should receive positive feedback. Otherwise, it should receive negative feedback:

$$H(s,a) = \begin{cases} +1 & \text{if } Q(s,a) \geq \alpha Q(s,a^*) \\ -1 & \text{otherwise} \end{cases} \tag{1}$$

The α increases over time (see Table 4) to encourage the agent to learn to choose better actions during training.

Hyperparameters and training procedure The hyperparameters used by evaluative feedback algorithms and experiments are shown in Table 4. To speed up training in the challenging `Placing-Real` task, we train the robot to move to the target area first, then train the gripper to release the ball. This reduces the amount of time spent on resetting.

3 Appendix: Preference Learning

Synthetic human for simulation experiments The goal for preference learning agents is to learn the reward function from preference signals. For simplicity, we assume that reward function $\mathcal{R}$ is a linear combination of state-action features [3]: $r(s_t, a_t) = \theta^T \phi(s_t, a_t)$. The reward features naturally overlap with abstract state dimensions in the scene graphs, since these scene graphs are designed to contain information that is critical to task success. The reward of a trajectory ξ is then:

$$r(\xi) = \sum_{(s,a)\in\xi} r(s,a) = \theta^T \sum_{(s,a)\in\xi} \phi(s,a) = \theta^T \Phi(\xi) \tag{2}$$

In simulation, we use a synthetic human (an oracle) who has access to the true reward weight θ. For every pair of queries(ξ, ξ'), the oracle calculates the true reward r and r' using Appendix Eq.2 with selected features. The oracle returns $q(\xi, \xi') = 0$ if $r(\xi) > r(\xi')$, and $q(\xi, \xi') = 1$ otherwise.

Hyperparameters The hyperparameters used by preference learning algorithms and experiments are shown in Table 5. In Algorithm 2 line 4, we query dimension by dimension. Empirically, we found that it is helpful to select multiple queries per dimension before moving to the next dimension. The number of queries per dimension can be seen in Table 5.

4 Appendix: Human Experiments

4.1 Evaluative feedback

Order of training trials Each human trainer participates in three training trials, one for each of the algorithms (EF-50%, EF-25%, DREF). The order of training matters here, since familiarity with

Parameter name	Value
# collected full trajectories for all tasks	120
`Lunar-Lander` # queries	60
`Reaching-Sim` # queries	60
`Placing-Sim` # queries	70
Simulation tasks # queries per dimension	10
`Reaching-Real` # queries	30
`Reaching-Real` # queries per dimension	6
`Placing-Real` # queries	30
`Placing-Real` # queries per dimension	10

Table 5: The list of hyperparameters for preference learning algorithms and experiments.

the tasks may affect the learning outcome. Therefore, we counterbalanced the order of the training trials. For six human trainers, their orders are:

1. DREF, EF-50%, EF-25%
2. DREF, EF-25%, EF-50%
3. EF-50%, DREF, EF-25%
4. EF-50%, EF-25%, DREF
5. EF-25%, DREF, EF-50%
6. EF-25%, EF-50%, DREF

Instructions to the human trainers Each human participant receives the following instruction at the beginning of the experiment:

> Thank you for participating in the human-robot study! In this study, you will train a robot to perform a given task, by providing guidance signals.
>
> [Task description]
>
> `Reaching`: The goal is to teach the robot to move its hand to the designated region on this table.
>
> `Placing`: The goal is to teach the robot to put the basketball inside the hoop.
>
> You will be watching a robot that is learning to do the task described above. The robot will ask for your guidance when **[it stops]**, and please provide your feedback. Please give positive feedback if observed behavior is desirable by pressing the **[U Key]**, negative feedback if observed behavior is undesirable by pressing the **[T Key]**, no feedback if you think the query does not make sense by pressing the **[Y Key]**.

Post-completion survey Each human participant receives the following post-completion survey after finishing all the training trials. These questions are designed to measure participants' overall experience training the robots (E, Question 1), perceived intelligence level (I, Question 2), and cognitive ease (C, Question 3).

> Thank you for your participation!
>
> 1. Please rate your overall user experience for the [three] trials
>   - Trial 1: (1-extremely unsatisfied, 5-extremely satisfied)
>   - Trial 2: (1-extremely unsatisfied, 5-extremely satisfied)
>   - Trial 3: (1-extremely unsatisfied, 5-extremely satisfied)
> 2. How intelligent was the robot's decision during the training?
>   - Trial 1: (1-extremely unintelligent, 5-extremely intelligent)
>   - Trial 2: (1-extremely unintelligent, 5-extremely intelligent)
>   - Trial 3: (1-extremely unintelligent, 5-extremely intelligent)
> 3. How difficult was it for you to provide signals?

    - Trial 1: (1-extremely difficult, 5-extremely easy)
    - Trial 2: (1-extremely difficult, 5-extremely easy)
    - Trial 3: (1-extremely difficult, 5-extremely easy)
4. Please leave any comments you have here.

4.2 Analyses

For Reaching, a repeated measures ANOVA was performed to compare the effect of three different algorithms (EF-25%, EF-50%, DREF) on the following variables:

- Total training time (T): There is a significant difference between three algorithms ($F(2, 10) = 21.62, p = .000$). DREF leads to significantly less training time than EF-50% ($p < .001$).
- No feedback count (NFC): There is a significant difference between three algorithms ($F(2, 10) = 8.52, p = .007$). DREF leads to significantly fewer "no feedback" responses than EF-25% ($p = .027$) and EF-50% ($p = .019$).
- Overall experience (E, measured by survey Q1): There is a significant difference between three algorithms ($F(2, 10) = 13.00, p = .002$). DREF is significantly better than EF-50% ($p = .012$) and EF-25% ($p = .017$).
- Perceived intelligence (I, measured by survey Q2): There is a significant difference between three algorithms ($F(2, 10) = 9.12, p = .006$). DREF is significantly better than EF-25% ($p = .012$) and marginally better than EF-50% ($p = .058$).
- Cognitive ease (C, measured by survey Q3): There is no significant difference among three algorithms ($F(2, 10) = 2.06, p = .178$).

For Placing, the same analysis was performed:

- Total training time (T): There is no significant difference among three algorithms ($F(2, 10) = 1.43, p = .285$).
- No feedback count (NFC): There is a significant difference between three algorithms ($F(2, 10) = 10.54, p = .003$). DREF leads to significantly fewer "no feedback" responses than EF-50% ($p = .012$).
- Overall experience (E, measured by survey Q1): There is a significant difference between three algorithms ($F(2, 10) = 7.71, p = .009$). DREF is significantly better than EF-50% ($p = .011$) and EF-25% ($p = .030$).
- Perceived intelligence (I, measured by survey Q2): There is a significant difference between three algorithms ($F(2, 10) = 8.57, p = .007$). DREF is significantly better than EF-50% ($p = .007$) and EF-25% ($p = .007$).
- Cognitive ease (C, measured by survey Q3): There is a significant difference between three algorithms ($F(2, 10) = 6.43, p = .016$). DREF is significantly better than EF-50% ($p = .044$) and EF-25% ($p = .007$).

4.3 Preference learning

Instructions to the human trainers Each human participant receives the following instruction at the beginning of the experiment:

> Thanks for participating in the human-robot study! In this study, you will watch recorded robot trajectories and provide preference signals.
>
> [Task description]
>
> `Reaching`: The goal is to teach the robot to move its hand to the designated region on this table.
>
> `Placing`: The goal is to teach the robot to put the basketball inside the hoop.
>
> Watch the clips and select the one in which better things happen. Only decide on events you actually witness. Don't worry about how the agent got into the situation it is in, just focus on what happens in the clip itself. If you find the first clip is better, then press the **[1 key]**. If you find the second clip is better, then press the **[2 key]**.

Post-completion survey Each human participant receives the following post-completion survey after finishing three trials (DRPL, random_fragment, DRPL-SS). These questions are designed to measure participants' overall experience training the robots (E, Question 1), perceived intelligence level (I, Question 2), and cognitive ease (C, Question 3). For Question 2, the final learned reward function is shown and explained to the participants.

Thank you for your participation!

1. Please rate your overall user experience for the [three] trials
   - Trial 1: (1-extremely unsatisfied, 5-extremely satisfied)
   - Trial 2: (1-extremely unsatisfied, 5-extremely satisfied)
   - Trial 3: (1-extremely unsatisfied, 5-extremely satisfied)
2. Does the final learned reward function match your expectation?
   - Trial 1: (1-extremely unmatched, 5-extremely matched)
   - Trial 2: (1-extremely unmatched, 5-extremely matched)
   - Trial 3: (1-extremely unmatched, 5-extremely matched)
3. How difficult was it for you to provide signals?
   - Trial 1: (1-extremely difficult, 5-extremely easy)
   - Trial 2: (1-extremely difficult, 5-extremely easy)
   - Trial 3: (1-extremely difficult, 5-extremely easy)
4. Please leave any comments you have here.

4.4 Analyses

For Reaching, a repeated measures ANOVA was performed to compare the effect of three different algorithms (random-fragment, DRPL-SS, DRPL) on the following variables:

- Overall experience (E, measured by survey Q1): There is no significant difference among three algorithms ($F(2, 10) = 0.45, p = .647$).
- Perceived intelligence (I, measured by survey Q2): There is a significant difference between three algorithms ($F(2, 10) = 30.45, p < .001$). DRPL is significantly better than random fragment ($p = .001$). DRPL-SS is also significantly better than random fragment ($p < .001$).
- Cognitive ease (C, measured by survey Q3): There is no significant difference among three algorithms ($F(2, 10) = 3.18, p = .131$).

For Placing, the same analysis was performed:

- Overall experience (E, measured by survey Q1): There is no significant difference among three algorithms ($F(2, 10) = 1.67, p = .237$).
- Perceived intelligence (I, measured by survey Q2): There is a significant difference between three algorithms ($F(2, 10) = 11.67, p = .002$). DRPL is significantly better than random fragment ($p = .001$).
- Cognitive ease (C, measured by survey Q3): There is no significant difference among three algorithms ($F(2, 10) = 0.14, p = .869$).

References

[1] Y. Zhu, J. Wong, A. Mandlekar, and R. Martín-Martín. robosuite: A modular simulation framework and benchmark for robot learning. *arXiv preprint arXiv:2009.12293*, 2020.

[2] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba. Openai gym. *arXiv preprint arXiv:1606.01540*, 2016.

[3] E. Bıyık, A. Talati, and D. Sadigh. Aprel: A library for active preference-based reward learning algorithms. *arXiv preprint arXiv:2108.07259*, 2021.