Supplementary Material

A. Pseudocode of the Algorithm

Algorithm 1 MILE: Model-based Intervention Learning in Iterative Setting

Notations
N: maximum deployment rounds, k: number of rollout episodes in each deployment round, l: number of batches, b: batch size, m: number of epochs in each learning round, α: learning rate, π_1^θ: initial policy, $\hat{\pi}_1^\xi$: initial mental model

for $i \leftarrow 1$ to N **do**
  $D^{i+1} \leftarrow \text{DEPLOYMENT}(\pi_i^\theta, D^i)$
  $\pi_{i+1}^\theta, \hat{\pi}_{i+1}^\xi \leftarrow \text{LEARNING}(\pi_i^\theta, \hat{\pi}_i^\xi, D^i)$

function DEPLOYMENT(π_θ, D)
  Collect rollouts w/ interventions $\tau_1, \ldots, \tau_k$
  $D' \leftarrow D \cup \{\tau_1, \ldots, \tau_k\}$
  return D'

function LEARNING($\pi_\theta, \hat{\pi}_\xi, D$)
  for m epochs **and** l batches each **do**
    Get the next mini-batch $\left(s^i, a_r^i, a_h^i, \nu^i\right)_{i=1}^b \sim D$
    Compute $\hat{\nu}(s^i; \theta, \xi) = p(\nu^i = 1 \mid s^i)$ based on Eq. (7)
    Compute $J(\theta, \xi)$ based on Eq. (12)
    Run in parallel:
      $\theta \leftarrow \theta - \alpha \nabla_\theta J(\theta, \xi)$
      $\xi \leftarrow \xi - \alpha \nabla_\xi J(\theta, \xi)$
  return π_θ, $\hat{\pi}_\xi$

B. Simulation Experiment Details

Depending on the action space, we trained suboptimal initial policies and various levels of expert policies for simulated humans using SAC or DQN [1, 2]. We train a different policy for each task. For each task, the same expert is used to intervene across all methods. To generate the mental models of simulated humans, we collected 100 rollouts of the initial policy and trained a BC agent on them. In all result plots, we display the mean and standard error over 3 seeds for each method.

Regarding the observation space, we use true world states, i.e. low-level states of the robot and the task-related object. This includes robot joint positions and the positions and orientations of relevant objects. Our action space consists of the change in Cartesian coordinates of the robot end effector and the gripper state. Each episode has a maximum of 1000 timesteps, with early termination upon success.

To compare our method with RLIF, we used RLPD as its backbone algorithm in the domains with continuous action spaces as it was done in the original paper [3]. For domains with discrete action spaces, we used DQN as the backbone of RLIF. We initialized the policy networks for all methods as the clones of the initial policy π_θ. For the offline demonstration ablation, we also initialized RLIF's replay buffer with those demonstrations.

C. Real Robot Experiment Details

In this experiment, we use image observations (captured from a USB webcam) along with the robot's end effector position. We keep the same action space as in the simulation experiments for real-world settings. The robot begins with the octagonal block in its gripper. The initial BC policy is trained using 120 human-collected trajectories, gathered with a Meta Quest 2 headset. At the beginning of real-world experiments, we show 3 trajectories to the users to make them aware of the robot's capabilities. We also warm-start the initial mental model using these same demonstrations. This process requires no supervision from the human and minimal effort since the human only needs to observe the robot.

D. Hyperparameters

We used a Multi-Layer Perceptron (MLP) with hidden dimensions of 256 in both the MetaWorld and real-robot experiments for all methods. In order to get image embeddings in real-robot experiment, we used a pretrained R3M model with ResNet50 architecture [4, 5]. To retain temporal information, we concatenate the states of the previous three timesteps with the current timestep. For the LunarLander experiment, we used an MLP with hidden dimensions of 64 for all methods. The network outputs an action for the current step.

The log-probabilities of the actions vary in scale between the policies used in different tasks. This necessitates adjusting the standard deviation of the normal distribution whose cumulative distribution function (CDF) is used in Equation 7, along with task-specific cost terms (parameter c), to calculate smoother rather than skewed intervention probabilities.

Table 1: Hyperparameters across different simulation and real-world tasks.

	LunarLander	**Button-Press**	**Drawer-Open**	**Peg-Insert**	**WidowX Peg Insertion**
MILE (Ours)					
Learning Rate	1e-5	1e-4	5e-4	1e-4	1e-5
Mental Model Hidden Dims	(64, 64)	(256, 256)	(256, 256)	(256, 256)	(256,256)
Dataset size in Offline Experiment	5 Trajectories	15 Trajectories	15 Trajectories	15 Trajectories	-
Number of Iterations	-	-	20	20	6
Episodes Per Iteration	-	-	1	1	3
Training Epochs per Iteration	-	-	300	300	500
Intervention CDF c	3	150	60	75	70
Intervention CDF σ	1	200	75	175	100
HG-DAgger					
Learning Rate	5e-6	5e-5	1e-4	5e-6	1e-5
Number of Iterations	5	5	5/20	5/20	6
Episodes Per Iteration	1	3	3/1	3/1	3
Training Epochs per Iteration	400	1000	1000/300	1000/300	500
RLIF					
Batch Size	64	256	256	256	-
Learning Rate	5e-4	3e-4	3e-4	3e-4	-
Discount	0.99	0.99	0.99	0.99	-
UTD Ratio	4	4	4	4	-
IWR					
Learning Rate	5e-6	5e-6	1e-4	1e-6	1e-5
Number of Iterations	5	5	5/20	5/20	6
Episodes Per Iteration	1	3	3/1	3/1	3
Training Epochs per Iteration	400	1000	1000/300	1000/300	500
Sirius					
Learning Rate	5e-6	5e-6	1e-4	1e-6	1e-5
Number of Iterations	5	5	5/20	5/20	6
Episodes Per Iteration	1	3	3/1	3/1	3
Training Epochs per Iteration	400	1000	1000/300	1000/300	500
All Methods					
Policy Hidden Dims	(64, 64)	(256, 256)	(256, 256)	(256, 256)	(256, 256)

References

[1] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. *International Conference on Machine Learning (ICML)*, 2018.

[2] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

[3] P. J. Ball, L. Smith, I. Kostrikov, and S. Levine. Efficient online reinforcement learning with offline data. In *International Conference on Machine Learning*, pages 1577–1594. PMLR, 2023.

[4] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016.

[5] S. Nair, A. Rajeswaran, V. Kumar, C. Finn, and A. Gupta. R3m: A universal visual representation for robot manipulation. In *Conference on Robot Learning (CoRL)*, 2022.