Supplementary Material

In this supplementary material, We first present details and visualizations on how we generated $\mathscr{D}_{\text{train}}$, the training distribution of human agents in assistive itch scratching in Sec. A. We then present the main algorithm pseudo-code for PALM in Sec. C. Next we provide implementation details of different methods in our main experiments in Sec. D and the ablation study of PALM in Sec. E and an evaluation of PALM with a second task, bed bathing assistance, in Sec. F.

A Generating Human Populations

To train our robot using sim2real, we would like to have a set of diverse environments. However, unlike single-agent domain randomization where we can vary environment parameters such as friction, in assistive tasks the environment entails a changing user policy. It is not obvious how to best generate a diverse population that captures user preferences, levels of disabilities, or movement characteristics. Generating human motions that realistically capture the variation observed in physical human-robot interaction has remained an unsolved challenge in robotics.

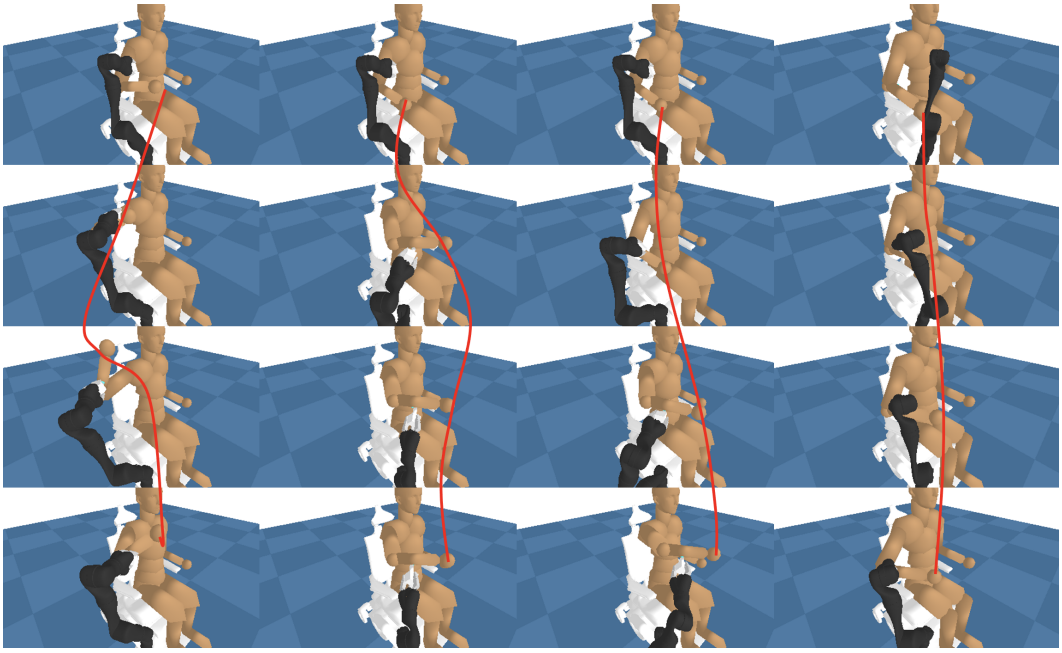

Figure 8: Visualization of humans generated of different activity levels. From left to right we apply action penalties $c_{\text{p}} = 0, 10, 30, 60$. Qualitatively, increasing the penalty results in the human taking more steady actions with less swinging motions. This results in the human being more likely to expose the itch spot for the robot to scratch, as opposed to scratching themselves.

Co-Optimization Prior works in robotic assistance [3, 24] have demonstrated that by optimizing for the same task objective, we can generate human and robot motions that coordinate towards the same goal, such as robot-assisted dressing. Further more, we can leverage reward engineering [25, 26] to generate a diverse set of motions.

To generate diverse population in itch scratching task, we explore two sources of diversities: (1) we assign different human action penalty c_{p}, where larger penalties lead to the human agent exerting less effort. In the simulation experiment we use $c_{\text{p}} = 3, 3.5, 4$.

(2) We simulate different itch positions on the human's arm and train co-optimized human and robot policies conditioned on them. This leads to qualitatively different strategies for the human and the robot. Note that this serves as the first step to understanding how different methods generalize since we never expect to be able to capture the diversity in humans perfectly. For training, we use Proximal Policy Optimization (PPO) to optimize human and robot policies in an interleaving fashion. Note that

we also keep the co-optimized robot policy and use it to obtain expert actions for assistive policy training (see Sec. 3.1 and supplement for full details).

Visualization Here we visualize trajectories from humans with different action penalties in Fig. 9. Note that higher penalties result in the human taking more steady actions of smaller magnitude.

B Comparison with other VAE baselines for sequential data

Note that our method relies on embedding human trajectories as sequential data into a latent space. Our implementation uses the final hidden state of RNN as the input to variational autoencoder. This is based on [27], which has been shown to be effective in embedding and generating sentences. Given that there are other different generative models for sequential data, our framework can be easily combined with them. In fact, wee believe coming up with a better model for human embedding is a future direction.

We hereby provide comparison with another different generative sequential model [28]. Different from [27] that uses only the final hidden state, they construct a latent space for every intermediate step in the sequential model. We keep all the experiment hyper-parameters the same, and concatenate the final hidden state with observation as input to the robot policy. We show the results in

Normalized Reward	Distribution	Our Method [27]	Baseline [28]	RNN
Assistive Reacher	IND	**0.72 ± 0.02**	0.66 ± 0.02	0.62 ± 0.02
	OOD	**0.38 ± 0.10**	0.11 ± 0.01	0.27 ± 0.09
Itch Scratching $\mathscr{D}^1$	IND	0.75 ± 0.01	0.45 ± 0.04	**0.79 ± 0.01**
	OOD	**0.48 ± 0.08**	0.32 ± 0.01	0.25 ± 0.08
Itch Scratching $\mathscr{D}^2$	IND	0.58 ± 0.03	**0.70 ± 0.03**	0.44 ± 0.01
	OOD	**0.49 ±.0.02**	0.31 ± 0.05	0.40 ± 0.02
Itch Scratching $\mathscr{D}^3$	IND	**0.34 ± 0.02**	0.23 ± 0.22	0.18 ± 0.01
	OOD	**0.25 ± 0.01**	0.13 ± 0.01	0.16 ± 0.01

Table 1: Comparison with RNN-VAE baseline [28]

C Algorithm

We present the main algorithm for PALM assume we have access to training and test distributions $\mathscr{D}_{\text{train}}$ and $\mathscr{D}_{\text{test}}$.

D Additional Training and Implementation Details

In our experiments in both the assistive reaching and assistive itch scratching, we use a recurrent network over a sliding window of 4-time steps, each of which is a concatenated vector of observation o_t, human action $a_{\text{H}t-i}$ and robot actions $a_{\text{R}t-i}$. For the current time step t, we use zero vector for $a_{\text{H}t}$ and $a_{\text{R}t}$. We set the latent space dimension to be four, and use a recurrent network with six layers. Our base policy network has four dimensions and hidden size of 100.

D.1 PPO and Behaviour Cloning

We need to train the base policy π_R and the latent space encoder $\mathcal{E}_\theta(z;\tau)$ jointly because they are interdependent — z is the input to π_R, and π_R decides the data distribution which leads to z. We simultaneously optimize the prediction loss in Eq. (2) and the policy loss using PPO [29] algorithm. See Appendix D for more training details. To amend for the instability of training with a population of humans, we leverage Behaviour Cloning, where we use expert robot policies obtained via co-optimization (see Appendix A) to supervise π_R on on-policy data. More specifically, we query the expert actions a_t^{exp} in a DAgger fashion[30] [3] during training, and optimize a_t^R to minimize deviation from it: $\mathcal{L}_{\text{BC}} = \sum_t ||a_t^{\text{exp}} - a_t^R||^2$. The overall policy optimization loss include three terms: latent prediction loss, PPO loss and the Behaviour Cloning loss: $\mathcal{L}_{\text{PALM}} = \mathcal{L}_{\text{pred}} + \mathcal{L}_{\text{PPO}} + \mathcal{L}_{\text{BC}}$.

[3] This ensures that we encounter no distribution shift at deployment time.

Algorithm 1 Prediction-based Assistive Latent eMbedding Training

Randomly initialize base policy π, encoder $\mathcal{E}_\phi$ parameterized by ϕ, decoder $\mathcal{D}_\theta$ parameterized by θ. Empty replay buffer D_1, window size w
for itr = $1, .., N_{\text{itr}}$ **do**
  for $i = 1, .., N_{\text{batch}}$ **do**
    Sample $\pi_{\text{H}_i} \sim \mathscr{D}_{\text{train}}$, optionally find pre-trained expert robot policy $\pi_{\text{E}i}$
    $o_0 \leftarrow$ env.reset()
    Initialize history $H \leftarrow \phi$
    for $t = 0, ..., T$ **do**
      Get latest w steps from H: $H_{-w} \leftarrow H[-w:]$.
      $z_t \leftarrow \mathcal{E}_\phi(o_t, H_{-w})$
      $a_{\text{H}t} \leftarrow \pi_{\text{H}_i}(o_0)$
      $a_{\text{R}t} \leftarrow \pi(o_t, z_t), a_{\text{e}t} \leftarrow \pi_{\text{E}i}(o_t)$
      $o_{t+1} \leftarrow$ env.step$(a_{\text{H}t}, a_{\text{R}t})$
      Store $(o_t, a_{\text{H}t}, a_{\text{R}t}, a_{\text{e}t}, H_{-w})$ in D_1
    end
  end
  for $j = 1, ..., N_{\text{opt}}$ **do**
    Sample a batch of $(o_t, a_{\text{H}t}, a_{\text{R}t}, H_{-\tau})$ from D_1
    Compute $\mathcal{L}_{\text{pred}}$ using Eq. (2), $\mathcal{L}_{\text{PPO}}$ using [29] and $\mathcal{L}_{\text{BC}} = \sum_t ||a_t^{\text{exp}} - a_t^R||^2$
    Optimize θ, ϕ, π for $\mathcal{L}_{\text{PALM}} = \lambda_{\text{pred}}\mathcal{L}_{\text{pred}} + \lambda_{\text{PPO}}\mathcal{L}_{\text{PPO}} + \lambda_{\text{BC}}\mathcal{L}_{\text{BC}}$
  end
end

Algorithm 2 Prediction-based Assistive Latent eMbedding Test Time Adaptation

Sample $\pi_{\text{H}} \sim \mathscr{D}_{\text{test}}$, Initialize history $H \leftarrow \phi$, Empty trajectory data τ
for $i = 0, .., N_{\text{adapt}}$ **do**
  $o_0 \leftarrow$ env.reset()
  for $t = 0, ..., T$ **do**
    Get latest w steps from H: $H_{-w} \leftarrow H[-w:]$.
    $z_t \leftarrow \mathcal{E}_\phi(o_t, H_{-w})$
    $a_{\text{R}t} \leftarrow \pi(o_t, z_t)$
    $o_{t+1} \leftarrow$ env.step$(a_{\text{H}t}, a_{\text{R}t})$
    Store $(o_t, a_{\text{H}t}, H_{-w})$ in τ
  end
  Compute $\mathcal{L}_{\text{pred}}$ using Eq. (2) on τ, $\theta \rightarrow \theta - \delta\nabla_\theta\mathcal{L}_{\text{pred}}(\mathcal{E}_\theta, \tau)$.
end

D.2 Hyperparameters

PALM Training We use $\lambda_{\text{PPO}} = 0.1, \lambda_{\text{BC}} = 1, \lambda_{\text{pred}} = 0.1$ in our experiments. We find that the behaviour cloning loss is essential for the Assistive Itch Scratching task. Under this hyperparameter setting, we train for 200 iterations. During each iteration, we collect 19,200 state-action transitions, which is evenly divided into 20 mini-batches. Each mini-batch is fed to the base policy and encoder for 30 rounds to compute the loss and error for back-propagation. We set the learning rate to be 0.00005.

For PALM prediction training, we use a three-layer decoder with hidden size 12 to predict the next human action from the hidden state from the encoder. To implement the KL regularization, follow the standard VAE approach. We use two linear networks to transform the encoder hidden state into μ and σ, which denote the mean and the standard deviation of the latent space. We then compute approximate KL divergence to normal distribution on this latent distribution.

PALM Test Time Optimization At test time, we roll out the trained robot policy and collect data with the same user for 25 iterations, or 2,500 time steps. This amounts to 150 seconds of wall clock time. We then optimize for the prediction loss (including KL regularization term) using learning rate of 0.0001 for one to five steps, and use the one with the lowest loss. We empirically find the

hyperparameters by doing the same process with humans from the training distribution, where we collect a mini training set and mini evaluation set both of 25 iterations. We use the mini training set to find the learning rate and use the evaluation set to ensure there is no over-fitting.

RILI/LILI Training We follow a similar approach to PALM, except that we learn to predict the next state o_{t+1} and scalar reward.

RMA Training We follow the two-phase training procedure in [7]. Note that we find it crucial in phase 2 to train the encoder with on-policy data, meaning that the regression data is collected by rolling out actions output by the "recurrent learner", not the trained network from phase 1. The phase 1 network is used simply for generating labels.

RNN/MLP Training. For RNN, we directly feed the hidden state of the recurrent encoder to the policy network. The architectural difference between RNN and PALM is that we do not concatenate the current observation o_t to the encoded output. To ensure that the policy has at least the same capacity as PALM, we use a base policy with the same number of parameters as in PALM.

E Ablation Studies

PALM Baselines	Reach	Itch $\mathscr{D}^1$	Itch $\mathscr{D}^2$	Itch $\mathscr{D}^3$
PALM test optim	**0.43 ± 0.13**	**0.51 ± 0.08**	**0.50 ± 0.02**	**0.29 ± 0.02**
PALM w/o test optim	0.38 ± 0.10	0.48 ± 0.08	0.49 ± 0.02	0.26 ± 0.01
No $\mathcal{L}_{\text{pred}}$	0.30 ± 0.05	0.50 ± 0.08	0.45 ± 0.02	0.25 ± 0.01
$c_{\text{KL}} = 0$	0.32 ± 0.08	0.47 ± 0.04	0.46 ± 0.01	0.23 ± 0.02
Frozen $\mathcal{E}$	0.24 ± 0.04	0.21 ± 0.06	0.15 ± 0.07	0.11 ± 0.05

Table 2: Normalized Reward on $\mathscr{D}_{\text{test}}$, standard deviation over 3 seeds.

We include ablation studies of PALM in the main experiment in Sec. 4, where we study the effect of test-time optimization, prediction loss, KL regularization and jointly training encoder $\mathcal{E}$ and policy π.

In the assistive reaching experiment, we observe that test-time optimization, $\mathcal{L}_{\text{pred}}$, KL regularization, and joint training all contribute to the OOD performance.

In the assistive itch scratching experiment, test time optimization and $\mathcal{L}_{\text{pred}}$ improve experiment results in all the settings. Applying KL regularization provides some gain in the complex distribution $\mathscr{D}_3$, but does not lead to improvement in simpler distribution $\mathscr{D}_1$ and $\mathscr{D}_2$.

F Additional Experiment: Bed Bathing Task

We further evaluate the performance of PALM in another assistive robotics task: robot-assisted bathing. This task is a modified version of the bathing task introduced in Assistive Gym [3]. In this task, we have a human lying on a tilted bed, with a robot mounted on the nightstand. The human can move their right arm, and there is an identified patch of skin to be cleaned. Unlike the original bed bathing task, where the entire right arm is covered in target points to be cleaned, we instead initialize a fixed size region of points to be cleaned at a uniform randomly selected location along the surface of the right forearm. The patch spans 10 centimeter along and 150 degrees around the forearm. Only the human knows the center position of the patch, and the robot must infer the location of the patch based on observations of the human motion. The points along the body are cleaned whenever the robot initiates contact with the spot using its end-effector and applies a positive normal force. The task reward is based on how many points are cleaned.

We generate synthetic humans by co-optimizing humans and robots conditioned on the centroid position of the region to be cleaned. During co-optimization, we adopt the formulation in [3] where both the human and the robot observe the centroid position along the human forearm. This helps induce collaborative behaviour for the human and robot policies, where we then use the resulting human as the synthetic population. During training, we blind of robot policy of the centroid position. We use the co-optimized robot (observes the centroid position) as the oracle for querying expert actions for Behaviour Cloning. We sample 18 humans from the training distribution, and save 6 humans from the held-out distribution as the out-of-distribution evaluation. We use the same hyperparameters as the itch scratching experiment.

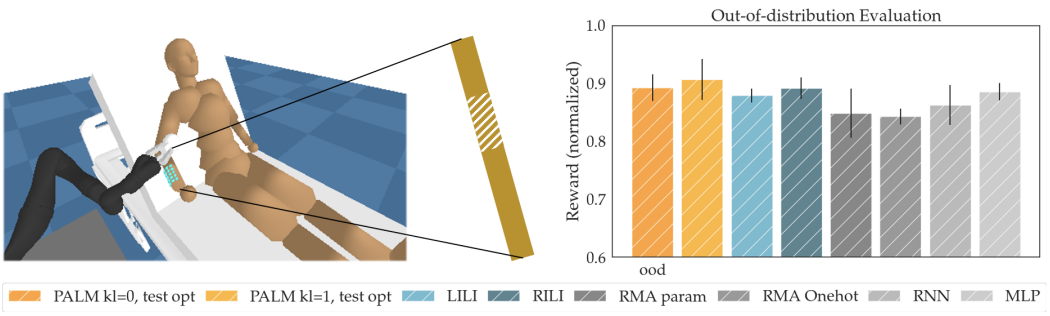

Figure 9: Visualization of the bed bathing task (left), where the human lies on a tilted bed with a table-mounted robot to clean an area on their arm. The area is random initialized (middle) on the forearm, where we hold out a quarter of the length as the held out distribution. The results of the held-out distribution is visualized on the right.

As we see in the results in Fig. 9, PALM achieves better out-of-distribution results compared to the baseline methods. We also observe that this performance gap is smaller than the itch scratching task. We believe this is due to the nature of the bed bathing task, where a robot controller that maintains contact with the human's forearm can be sufficient for solving the task if the human policy learns to move and rotate their forearm accordingly to help the robot.