A Environment Details

In this section, we provide further details of the environment, offline data, and target tasks for our real-world and simulated experiments.

A.1 Real-World Environment

Environment. We use a similar robotic platform as in Ebert et al. [8]. The platform includes a 6-DoF WidowX robot with a 1-DoF parallel-jaw gripper installed in front of a kitchen-themed tabletop manipulation environment. A LogiTech webcam is mounted over-the-shoulder to capture 128×128 RGB images as observations. The action is 7-dimensional which includes the 6 DoF gripper pose and 1 continuous value which controls the finger status. We use the toy kitchen 2 from Ebert et al. [8] as the target scene, where all of the target tasks take place in.

Offline data. As shown in Fig. 6, we use the bridge dataset from [8] for the real-world experiments, which is a large and diverse dataset of robotic behaviors collected in various scenes with different objects and illuminations. The trajectories are collected through tele-operation using virtual reality equipment. The training is conducted on 11,980 trajectories, which is comprised of 2,061 trajectories from toy kitchen 2 and 9,919 trajectories from the other scenes of the bridge data. These training data include picking and placing objects into different destinations, turning a faucet, tilting a pot upright, etc. The objects used in the target tasks are held out from the offline data.

Figure 6: **Example trajectories of the offline data for real-world experiments.** We use the Bridge Dataset [8] for our real-world experiments, which is composed of demonstration trajectories collected through tele-operation from 10 different scenes with a variety of objects.

Target tasks. In each target task, a desired goal state is specified by a 128×128 RGB image (same dimension as the observation). The robot is tasked to reach the goal state by interacting with the objects on the table. Task success for our evaluation is determined based on the object positions at the end of each episode (this metric is not used for learning). We design three target tasks that require multi-stage interactions with the environment to complete. The objects in these target tasks are unseen in the target domain in the offline data. The episode length is 75 steps in the real world.

A.2 Simulated Environment

Environment. Our simulated experiments are conducted in a table-top manipulation environment with a Sawyer robot. At the beginning of each episode, a fixed drawer and two movable objects are randomly placed on the table. The robot can change the state of the environment by opening/closing the drawer, sliding the objects, and picking and placing objects into different destinations, etc. At each time step, the robot receives a 48 x 48 RGB image as the observation and takes a 5-dimensional continuous action to change the gripper status through position control. The action dictates the

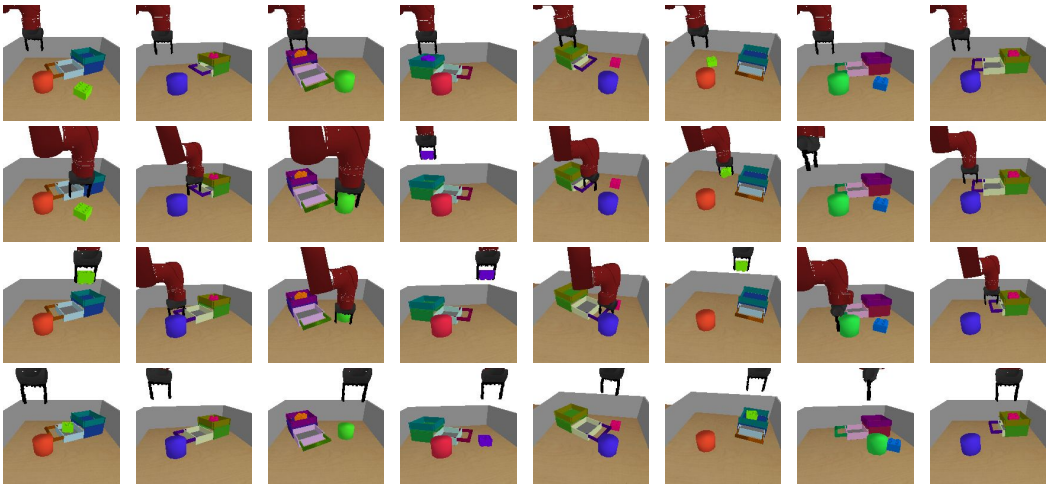

Figure 7: **Example trajectories of the offline data for simulated experiments.** Our offline dataset for simulated experiments are collected by a scripted policy using privileged information from a variety of scenes and camera viewpoints. The collected trajectories involve diverse interactions with the environment such as picking, placing, pushing, opening drawer, and closing drawer.

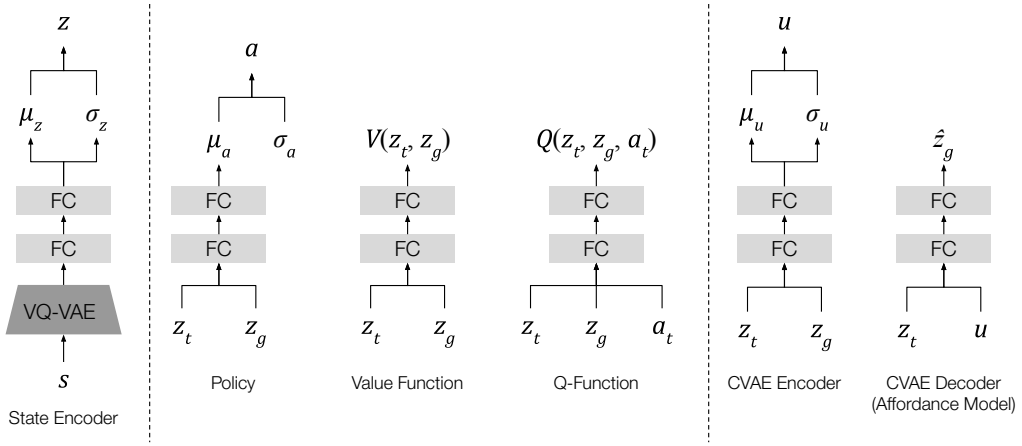

Figure 8: **Network architectures.** We show the architectures of the state encoder, the policy network, the value function, the Q-function, the CVAE encoder, and the CVAE decoder (affordance model).

change of the coordinates along the three axes, the change of the rotation, and the status of the fingers. The simulated environments are implemented with a real-time physical simulator [62].

Offline data. Example trajectories of our offline data in the simulation is illustrated in Fig. 7. We collect 12,000 trajectories using a scripted policy to perform primitive behaviors such as picking, placing, pushing, opening, and closing. The scripted policy utilizes privileged information such as the ground truth object poses and the gripper pose. These trajectories are collected from environments of diverse camera viewpoints and object appearances. In the offline dataset, we hold out interactions with the drawer (opening and closing) in the target scene, which is an essential type of behaviors involved in all the target tasks.

Target tasks. We design three target tasks that require strategically stitching together multiple behaviors, including opening/closing the drawer and other types of interactions with objects in the scene. In each target task, a desired goal state is specified by a 48×48 RGB image (same dimension as the observation). The robot is tasked to reach the goal state by interacting with the objects on the table. The task success is determined based on the object positions at the end of each episode. The episode length is 400 steps in simulation.

B Implementation Details

Network Architectures. We illustrate the network architectures of the state encoder, the policy, the value network, the Q-network, and the affordance model in Fig. 8. The state encoder projects the images into 128-dimensional representations. The state encoder is implemented with two 128-dimensional fully-connected (FC) layers on top of a VQ-VAE [18] backbone. The VQ-VAE backbone is trained on the offline dataset using the reconstruction loss as in [18] and its weights are fixed during the offline pre-training and online fine-tuning in FLAP. The state encoder outputs the mean μ_z and standard deviation σ_z for sampling the lossy representation z. The policy network, the value network, and the Q-network are implemented with 128-dimensional FC layers. Inputs to these networks are concatenated and then fed to the FC layers. The policy network produces the mean μ_a and a state-independent standard deviation σ_a is jointly learned to form the action distribution. The affordance model operates in an 8-dimensional latent space, and both its encoder and decoder are implemented by 128-dimensional FC layers. The affordance model is trained to predict subgoals of $\Delta t = 30$ in simulation and $\Delta t = 20$ in the real world using the training protocol from [7].

Learning. We use Implicit Q-Learning(IQL) [11] as the underlying RL algorithm. To avoid evaluating actions outside of the offline dataset, IQL performs a modified TD update by approximating an upper expectile of the distribution over values. We use Adam optimizer with a learning rate of 3×10^{-4} and a batch size of 128. The Lagrange multipliers in Eq. 2 and Eq. 4 are chosen through grid search and are set to be $\alpha = 0.01$ and $\beta = 0.1$. During offline pre-training, we train the model on the previously collected datasets with only relabeled goals [38]. While during online fine-tuning, we use 30% ground truth goals and 70% relabeled goals. In each batch, we sample 60% of the samples from the offline dataset and the rest 40% from the online replay buffer following the practice of Fang et al. [7]. We use 1 TITAN RTX GPU with 24GB of memory and 1 Intel Xeon Skylake 6130 CPUS with 48GB of memory during training. The pre-training takes around 10 hours in total and the fine-tuning takes around 2 hours for each target task.

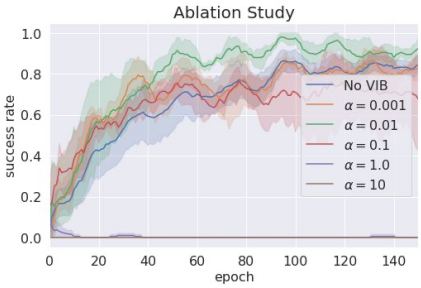

Figure 9: **Ablation Study.** We analyze the trade-off between the variational information bottleneck and the RL objective by sweeping the hyperparameter α in Eq. 2.

Planning. Model predictive path integral (MPPI) [12] is used for planning the subgoals as in Fang et al. [7]. We first sample 1,024 sequences of latent codes from the prior distribution $p(u)$ and chooses the optimal plan through the planning procedure described in Sec. 3.2. Then we refine the plan through 5 MPPI iterations. In each iteration, we add Gaussian noises scaled by 1.0, 0.5, 0.2, 0.1, 0.1 to the selected sequence of latent codes from the previous iteration. In Eq. 6, we set the weight $\eta_1 = 1$, $\eta_2 = 1$ and $\eta_3 = 0.1$. The planner produces 4 subgoals and the last subgoal is replaced with the final goal. The planner switches to the next subgoal in two conditions: (1) when the current subgoal is reached (2) if the current subgoal is not reached within a time budget of h steps. We set $h = \Delta t$ to spare a more generous time budget for the trained policy to reach each subgoal, where Δt is the time interval for training the affordance model described in Sec. 3.2.

C Ablation Study

We conduct ablation studies on the simulated target task C to analyze how the trade-off between the variational information bottleneck and the RL objective affects generalization. We sweep the weight α from 0.0 to 10.0 in Eq. 2 and analyze the performance of the fine-tuned policy in the target task. As shown in Fig. 9, the optimal fine-tuning performance is achieved with $\alpha = 0.01$. When α is too small, although the RL loss is lower on the offline dataset, the pre-trained policy does not generalize well in the target task. When α is too large, the pre-trained policy performs poorly, since the lossy representation loses too much information.

D Qualitative Results

Task execution. In Fig. 10, we show example trajectories of using the policy fine-tuned by FLAP to execute the target tasks. In the real world, our policies successfully control the WidowX robot to move pot from stove to stove, move colander onto stove and then drop object into colander, and then place sushi on plate and drop knife in pan. In simulation, the Sawyer robot is controlled to open/close the drawer and slide the cylinder. Some of these tasks require the robot to strategically interact with the objects in the scene in a specific sequential order. For example, in Task C in simulation, the robot needs to clear obstacle in front of the drawer before opening the drawer.

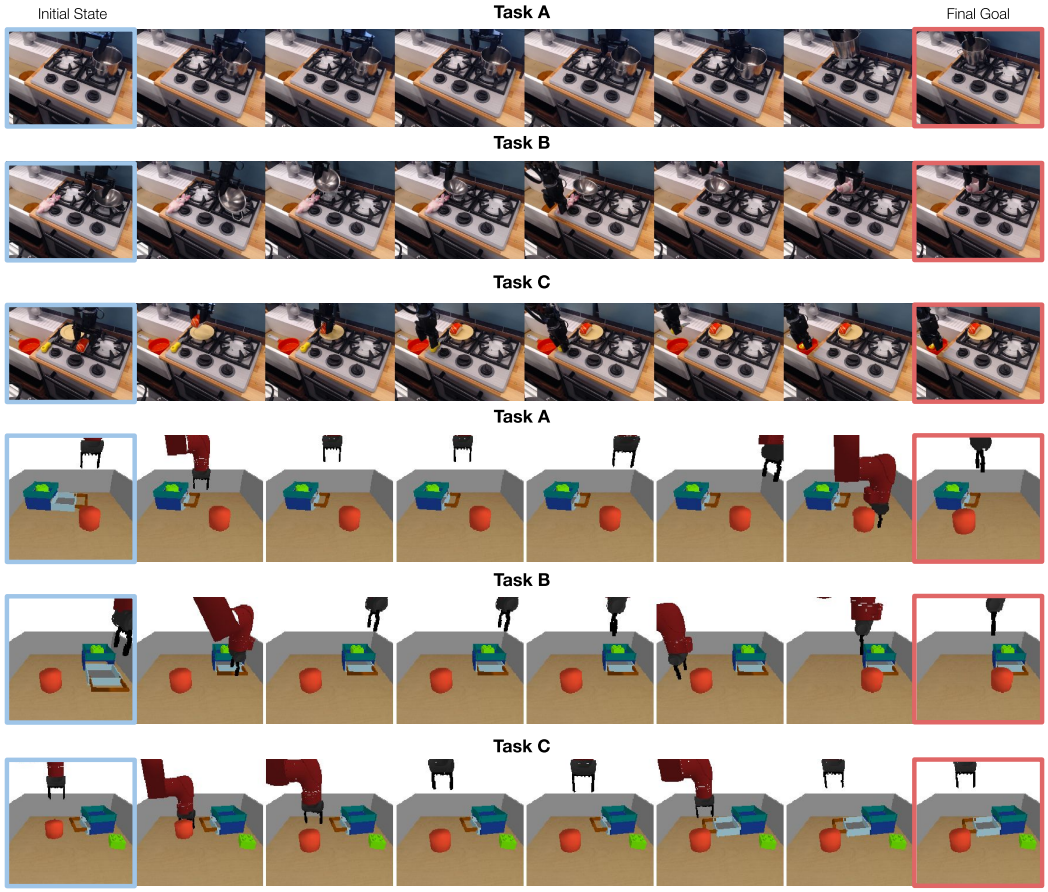

Figure 10: **Task Execution.** The three target multi-stage manipulation tasks in the real world and in simulation. Each row shows frames from a single episode, in which the initial state and the final goal are marked in blue and red.

Planned subgoals. While it is hard to directly visualize subgoals planned in the learned lossy representation space, we analyze their characteristics in Fig. 11. Same as in Sec. 4.3. we project image observations in the target scene using t-SNE [17]. A trajectory of executing the real-world Task B using the fine-tuned policy it visualized on this t-SNE plot. The orange dots mark the computed lossy representations of the initial state s_0 and the final goal g of the task. The subgoals planned in the lossy representation space $\hat{z}_1, ..., \hat{z}_3$ are shown as blue dots on the plot. The observed images at each step during task execution are plotted as the blue curve. Although the lossy representations cannot be directly reconstructed to the image space, we compute their distances to the computed representations of the observed images at each step. The closest observation to each planned subgoal is shown on the plot. As we can see, the planned subgoals are close to the executed trajectory. The first subgoal commands the robot to reach to the colander. The second subgoal commands the robot to place the colander onto the stove on the left. In the third subgoal, the robot gripper aims to pick up the toy from the table.

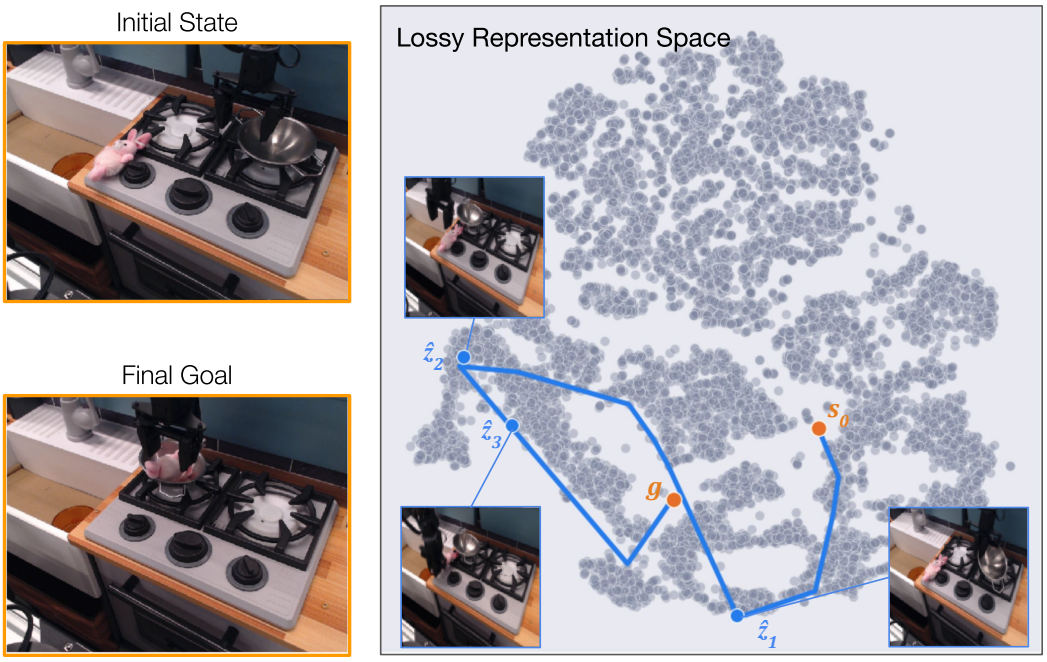

Figure 11: **Visualization of the planned subgoals.** The orange dots indicate the initial state and the final goal. The blue curve indicates the executed trajectory. The blue dots indicate the planned subgoals in the lossy representation space. The image observations that are closest to each planned subgoal is shown on the plot.