

A Appendix

A.1 Factorized Contextual Markov Decision Process

As introduced in Sec. 3, we model our problem as a contextual Markov Decision Process (CMDP) and propose a novel factorization of it into two subproblems that can be addressed with two collaborative policies trained separately: an *information-seeking (IS)* policy, π_{IS} , that searches and provides context to a second policy, an *information-receiving (IR)* policy, π_{IR} , that consumes the context and takes reward-seeking (manipulation) actions based on it. In the factorized CMDP (fCMDP), we assume that the trajectories generated by the optimal information-receiving policy, $[a^*(t=0), \dots, a^*(T)]$, will achieve maximum return in the CMDP, R_c , due to the values in some of the dimensions of the action vector, independently of the values in others, and that these actions can be inferred from a subspace of the observation space. Similarly, we assume that the trajectories generated by an optimal information-seeking policy will reveal the true context because of the values in some of the dimensions of the action vector, independently of the values in others, and that this true context can be inferred from a subset of the observation space. We factorize the action and observation spaces of each agent in the fCMDP based on what action dimensions are necessary to control to achieve the task reward vs. to reveal information, such that: $\pi^{IS} : O_{IS} \rightarrow A_{IS}$ and $\pi^{IR} : O_{IR} \times C \rightarrow A_{IR}$, with $O = O_{IS} \cup O_{IR}$. Therefore, only IS actions can lead to IS observations with information to infer the right context, and IR actions can change the state toward the overall task goal. The context can be inferred by the IS agent by mapping its observation(s) into a context with a given or learned function, $f_c : O_{IS} \rightarrow C$. However, for this function to map to the right context (i.e. the one that leads the IR policy to accumulate the highest return), the IS policy needs to take the right actions that reveal an information-rich observation. There is no constraint in the overlap between action and observation spaces of both policies but our method performs best in cases where there is little or no overlap.

This factorized CMDP (fCMDP) matches a natural factorization into agents with different action and observation spaces, e.g., when the IS agent controls the head of a humanoid and the IR agent controls the arm and manipulates the environment, or when the IS agent is a navigating agent scouting the environment for an IR agent that waits for the contextual information to act in front of a table.

A.2 IS Policy Training Procedure

Alg. 1 includes the pseudo-code of the DISaM’s Information Seeking (IS) agent. Specifically, DISaM trains the IS agent by iterating between the IS policy optimization loop and the encoder optimization loop. The policy optimization uses on-policy data whereas the encoder optimization utilizes data sampled from a replay buffer aggregated during IS policy training. The IR agent takes action if and only if it can reconstruct the true IR actions based on the information c_{IS} provided by the IS agent.

A.3 DISaM Deployment Procedure

Alg. 2 includes the pseudo-code of the DISaM’s deployment solution. During deployment, DISaM relies only on environmental observations to make decisions (no oracle or ground truth); the uncertainty of the IR policy is compared against a hyperparameter, δ , to determine when to query the IS agent or when to execute IR’s actions (see Fig. 5). To compute the uncertainty, DISaM samples n contexts $\{c_t^i\}_{i=1}^n$ from the ensemble of trained encoders (see Sec. 4.3), E_ϕ , and conditions the IR agent on each of them to generate n action distributions, $\{\pi^{IR}(o, c_{IS}^i)\}_{i=1}^n$. DISaM then computes the average KL-divergence between each pair of action distributions as a measure of uncertainty. Due to the difference in the scale of the KL-divergence, the threshold δ needs to be adapted to each action space. However, we found our method relatively robust to this parameter: we simply use $\delta = 0.5$ for all skill-based tasks (discrete action space), and $\delta = 1e5$ for visuomotor tasks (continuous action space), working well across different tasks.

Algorithm 1 Iterative Optimization for the Information Seeking Agent

```
1: Initialize Information-seeking policy  $\pi_{\theta}^{IS}$ , observation encoder  $E_{\psi}$ , Encoder Replay Buffer  $\mathcal{B}_{\mathcal{E}}$ ,  
   Rollout Buffer  $\mathcal{B}$ , switch threshold  $\mathcal{T}$   
2: for  $i$  in  $1, 2, \dots, K$  do  
3:   Empty  $\mathcal{B}$   
4:   Recieve initial observations  $o_{IS}, o_{IR}$   
5:   for  $i$  in  $1, 2, \dots, K_{\pi}$  do  
6:      $o_{IS}, o_{IR} \leftarrow \text{CollectRollout}(o_{IS}, o_{IR})$   
7:   end for  
8:   Optimize  $\pi_{cam}$  on  $\mathcal{B}$  with PPO objective  
9:   for  $i$  in  $1, 2, \dots, K_{enc}$  do  
10:     $o_{IS}, c_{GT} \sim \mathcal{B}_{\mathcal{E}}$  ▷ Sample a batch from replay buffer  
11:    UpdateEncoder( $o_{IS}, c_{GT}$ )  
12:  end for  
13: end for  
14:  
15: procedure COLLECTROLLOUT( $o_{IS}, o_{IR}$ )  
16:   Take actions  $a_{IS} \sim \pi_{\theta}^{IS}(o_{IS})$  and obtain  $o'_{IS}, c_{GT}$  from the environment  
17:    $c_{IS} \leftarrow E_{\psi}(o'_{IS})$   
18:    $r = \max(1 - \text{LossFunc}(o_{IR}, c_{IS}, c_{GT}), -1)$   
19:   add  $(o_{IS}, a_{IS}, r, o'_{IS})$  to  $\mathcal{B}$   
20:   add  $(o'_{IS}, c_{GT})$  to  $\mathcal{B}_{\mathcal{E}}$   
21:   while  $\text{LossFunc}(o_{IR}, c_{IS}, c_{GT}) < \mathcal{T}$  do  
22:     Take actions  $a_{IR} \sim \pi^{IR}(o_{IR}, c_{GT})$  and obtain  $o_{IR}$  from the environment  
23:   end while  
24:   return  $o'_{IS}, o_{IR}$   
25: end procedure  
26:  
27: procedure UPDATEENCODER( $o_{IS}, c_{GT}$ )  
28:    $c_{IS} \leftarrow E_{\psi}(o_{IS})$   
29:    $\psi = \text{argmin}_{\psi} \text{EncoderLoss}(c_{IS}, c_{GT})$  ▷ Optimize  $E_{\psi}$  using gradient descent  
30: end procedure  
31:  
32: procedure LOSSFUNC( $o_{IR}, c_{IS}, c_{GT}$ )  
33:   return  $\text{Distance}[\pi^{IR}(o_{IR}, c_{IS}), \pi^{IR}(o_{IR}, c_{GT})]$   
34: end procedure
```

536 A.4 Using Language as Context

537 In several of our tasks in simulation, the context is specified using a language instruction that speci-
538 fies the goal for the task. Each task stage is specified with a different instruction; the set of possible
539 language instructions for an entire task results from the Cartesian product of possible instruction
540 for each stage. Thus, the language instructions used in the Cooking (sim) task include $\{\text{"Lift$
541 $\text{up the bread"}\} \times \{\text{"Cook for a short amount of time"}, \text{"Cook until it is well-}$
542 $\text{done"}\} \times \{\text{"Place the pot on the red region"}, \text{"Put the pot on the green area"}\}$. The instructions
543 used in the Walls (sim) task include $\{\text{"Pick up the blue cube"}, \text{"Lift the wooden cube"}\} \times \{\text{"Place$
544 $\text{the cube on the red region"}, \text{"Put the cube on the green area"}\}$. When the environment is initialized
545 we select the sentences that correspond to the correct instructions, concatenate them to form a single
546 sentence, and process them with CLIP [72] to generate a language feature that acts as contextual
547 information about the goal of the task.

548 Since language embeddings are continuous vectors, we model them as Gaussian mixture models
549 using Mixture Density Networks (MDNs) [73] as the prediction head for E_{ϕ} and use the negative
550 log-likelihood loss to train E_{ϕ} . This is different from how we model and train the encoder when
551 contexts are one-hot vectors. In that case, while using the same backbone network of E_{ϕ} as language

Algorithm 2 DISaM Deployment

```

1: procedure DEPLOYAGENT( $\pi_{\theta}^{IS}, \pi^{IR}, E_{\psi}$ )
2:   Recieve initial observations  $o_{IS}, o_{IR}$ 
3:   repeat
4:      $\{c_t^i\}_{i=1}^n \leftarrow E_{\psi}(o_{IS})$ 
5:      $A_{\text{dist}} \leftarrow \{\pi^{IR}(o_{IR}, c_{IS}^i)\}_{i=1}^n$ 
6:     Uncertainty  $u \leftarrow \text{PairwiseKL}(A_{\text{dist}})$ 
7:     if  $u < \delta$  then
8:       Take action  $a_{IR} \sim A_{\text{dist}}$ 
9:       Receive observations  $o_{IS}, o_{IR}$ 
10:    else
11:      Take action  $a_{IS} \sim \pi_{\theta}^{IS}(o_{IS})$ 
12:      Receive observations  $o_{IS}, o_{IR}$ 
13:    end if
14:  until episode done
15: end procedure

```

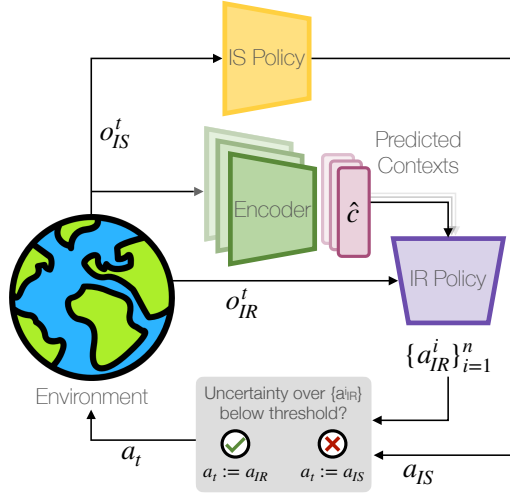


Figure 5: **Deployment of DISaM.** During deployment, DISaM takes actions based only on environmental observations. When the uncertainty of IR’s next action is low, DISaM follows the IR’s policy actions to accomplish the overall manipulation task goal; when the uncertainty is high, DISaM follows IS’s policy to reveal and obtain useful contextual information from the environment that helps to reduce IR’s uncertainty.

552 contexts, we replace the MDN head with a categorical distribution and train E_{ϕ} using the cross-
 553 entropy loss.

554 A.5 Stage Wise Success Rates and Analysis

555 In our experiments, three tasks consist of multiple stages leading to several phases of rising un-
 556 certainty, exploration with IS, and decreasing uncertainty, which are dynamically handled by our
 557 deployment procedure (Appendix A.3). Specifically, a stage may arise when the entire context can-
 558 not be reconstructed using a single information-seeking observation, o_{IS} , triggering a phase in the
 559 dual policy execution where the IS policy searches and provides the IR agent with the partial context
 560 necessary to complete the current stage before moving to the next one. For example, in the Cooking
 561 (sim) task, the side dish to prepare, the cooking time, and the serving location are determined by
 562 the IS policy looking over to different locations: the dinner, the clock, and the serving region respec-
 563 tively; each of these parts is denoted a “stage” of the task. In Walls (sim), the stages are picking
 564 and placing, with the block to pick determined by an identical block placed behind the wall on the

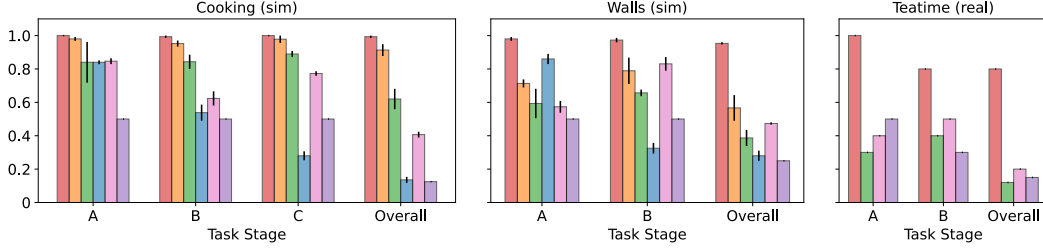


Figure 6: **Stage-wise success rates.** Success rates for each method are plotted for each stage of the multi-stage tasks: Cooking (sim), Walls (sim), and Teatime (real).

left and the placing region determined by the color of the block on the right. Finally, in the *Teatime* (real) task, the agent must look at the time to determine the appropriate drink, then the person to determine the serving location. Note that knowledge of these stages is not required by DISaM, which automatically infers the necessary information to collect at each stage through the intrinsic reward of minimizing the IR policy loss.

We evaluated DISaM, ablations, and baselines on each stage in isolation in addition to the overall task performance reported in the main text. Fig. 6 shows the success rates for each method for each stage of the multi-stage task as well as the overall success rate. In the simulation tasks, we performed 50 policy rollouts per seed and calculated the success rate of each stage as (# of stage successes)/(# times stage reached) to avoid accumulating in the results the effect of the previous stages. In the real environment, 10 rollouts were performed by resetting the environment to the initial state for each stage and evaluating the agent. The methods that receive task reward information (DISaM (task reward) and Full RL) use a reward provided at the end of each stage rather than the sparse task reward.

DISaM (language) and DISaM (task reward) are generally competitive in the individual stages for Cooking (sim), but the stage-wise failure rates compound resulting in lower overall success. On the more challenging Walls (sim) and Teatime (real), the variants are less competitive in each stage. Across all tasks, the baselines Full RL, Random Cam, and Sampled Context generally perform poorly in all stages, with the exception of Random Cam succeeding in some of the simpler stages (e.g., Cooking (sim) stage A, Walls (sim) stage B). Full RL achieves consistent success only in the first stage of Cooking (sim) and Walls (sim), perhaps due to experiencing these early stages more frequently during training.

A.6 Real World Training leveraging Synced Simulation

Performing reinforcement learning training procedures in the real world is challenging. Among the known challenges [74], two are significantly problematic: safety of the agent and the environment, and the need for constant resets (usually by a human). In our real-world training procedure we propose solutions for these two challenges, leading to an autonomous and safe training process: we developed a mixed sim-real training setup for training the IS policy and observation encoder using DISaM in real-world tasks.

Firstly, we create a sim version of the real-world task and collect data in both environments to train a corresponding IR policy. For IS policy training (see Fig. 2 and Alg. 1), we train the IS policy and encoder in the real world and generate the real-world context vectors but we exploit the simulator to interpret the context vector with the IR policy and provide rewards to the real IS policy. This eliminates the need to roll out the real IR policy, minimizing the resets and human supervision required during training. Once the IS policy is trained, we can then deploy it with the real IR policy following the system depicted in Fig. 5 and described in Alg. 2.

When the domains of the training and testing IR policy are different, we cannot use the IR observations to inform the IS policy decisions. One limitation arising from this is that we cannot do tasks

requiring multiple stages: the IS policy cannot utilize the IR observations to learn which information is being queried. Hence, in the `Teatime (real)` task, we separately train and evaluate the IS and IR policy in the two separate stages of the task, but consider them as a single task with two stages in our evaluations as they execute consecutively. Another issue that results from training the IS policy in the real world is the constant need to update the information in the environment that represents context, e.g., the time on the clock or the person sitting next to the table. Since in this work, we focus on the visual cues to represent the context, we choose to portray this information on screens, which allows us to programmatically update the context altering easily the environment whenever the visual cue needs to be updated (as described in Appendix A.9).

While in the simulation we use separate cameras for the IR and IS agents, in the real-world tasks both agents share the same camera, mounted on the head of a PAL Tiago++ [69]. We enable this by choosing a resting head position for the IR agent and resetting back to it whenever we switch to the IR policy deployment phase from the IS policy deployment phase. This is analogous to installing a second camera stationary pointing at the manipulation area. Additionally, during the IS deployment phase, we store the latest observation from the IR deployment phase and use it to calculate the uncertainty of the IR policy.

A.7 Baselines

Below we provide more information and implementation details of the baselines,

DISaM (reward): An ablation of our method which uses task rewards to train the IS agent instead of our proposed intrinsic reward. This baseline is used to test if a high-performance IS policy can be learned just by using task rewards. Instead of using sparse task rewards, which are difficult to learn from, we use hand-designed rewards at the end of key stages. While this stage-wise reward requires some domain knowledge to design, we found it to be important for the learning of the IS policy. Specifically, after completing a stage the reward obtained is +10 in simulation environments and +5 in real-world settings. For all other timesteps, the reward is -0.1. Across all our experiments we use the same hyperparameters and network architectures as we used for DISaM.

Full RL: A reinforcement learning baseline that jointly optimizes the IR and IS agents. The policy architecture is a shallow convolutional neural network to process image inputs followed by an MLP, that takes observations of both IR and IS agents as inputs and outputs an action over the combined action space of the two agents. The policy is trained using PPO [71] with the same stage-wise rewards as described above in the DISaM (reward) section.

Random Cam: A baseline that replaces the trained IS agent in DISaM with an agent that performs random movements but utilizes the trained encoder E_ϕ to encode the context and hands control back to the IR policy when it's uncertainty is lower than the prefixed threshold δ (similar to our test time protocol). Instead of sampling actions uniformly and randomly at every step, we perform weighted sampling by putting more weight on the last action performed, enabling the policy to explore a larger proportion of the state space. Specifically, with probability $0.5 + \frac{1}{n_actions}$, the action of the policy stays the same as in the previous timestep.

Sampled Context: We sample the context vector from a prior probability distribution over contexts and use it to produce IR actions using an IR policy trained with behavior cloning. In all our settings, the prior distribution consists of a uniform probability distribution over all possible contexts.

A.8 Comparing Visuomotor and Skill-based IR policies

The training procedure between visuomotor and skill-based IR policies is similar except for some minor differences that we note here. Primarily the distance metric \mathcal{L} in Equation 1 varies between the two IR policies. When the IR policy is skill-based, \mathcal{L} is the cross-entropy loss between the action distribution induced by the true and the predicted contexts. When the IR policy is visuomotor, we measure the negative log-likelihood of sampled predicted actions under the distribution of true actions. To demonstrate that the performance of DISaM is not affected by the type of IR policy used,

we compare DISaM’s performance trained with a skill-based (DISaM (skill)) and a visuomotor (DISaM (motor)) IR policy on the Walls (sim) task. (DISaM (skill)) achieves an overall success rate of 47.67 ± 0.58 and (DISaM (motor)) achieves an overall success of 47.33 ± 0.58 , measured over 50 rollouts across 3 seeds. As we can see, the overall performance of the system stays consistent across both forms of the IR policy.

A.9 Tasks

In Fig.7 we visualize rollouts in all tasks and in the following we describe the details of IR and IS, observation and action spaces in each task.

Walls (sim): Shown in Fig.7a, the IR agent is required to pick a block and place it on one of the serving regions. The color of the block to pick is determined by an identical block placed behind a wall on the table on the left and the color of the block on the table on the right corresponds to the correct serving region. Here the IR agent is the robot hand that uses the images from the eye-in-hand camera along with its proprioceptives as observations and outputs a skill ID to choose between pick, place and go-near actions. The IS policy observes the scene from a floating camera with discrete actions that allow the camera to pan left/right, tilt up/down, move forward/backward/left/right by some fixed amount.

Assembly (sim): Shown in Fig.7b, the IR agent is required to assemble the screw in the correct colored peg. The color of the correct peg is determined by a similarly colored block placed inside one of the drawers on the table on the right. Here the IR agent is the robot hand that uses the images from the eye-in-hand camera along with its proprioceptives as observations and outputs a skill ID to choose between pick-place and go-near actions. The IS policy observes the scene from a floating camera and also can control the second robot arm near the drawers to interact with them. The IS policy has a discrete action space that allows it to pan and tilt the camera, along with controlling the second robot arm to open the drawers and pick/place the blocks.

Cooking (sim): Shown in Fig.7c, the IR agent is required to cook a meal. The proper meal to prepare is determined by the dish being prepared, the cook time depends on the timer and the serving region is determined by the check mark. Here the IR agent is the robot hand that uses the images from the eye-in-hand camera along with its proprioceptives as observations and outputs a skill ID to choose between pick-place and go-near actions. The IS policy observes the scene from a floating camera with discrete actions that allow the camera to pan left/right, and tilt up/down.

Teatime (real): Shown in Fig.7d, the IR agent needs to determine the time of day to decide on the beverage to serve and look at the person at the table to choose where to place the beverage. Here the IR agent is a Tiago++ robot that uses the images from its head camera along with its proprioceptives as observations and outputs a continuous delta action in the Cartesian space to control its hand. The IS policy observes the scene from Tiago’s head camera as well (see Appendix A.6) with discrete actions that allow the camera to pan left/right, and tilt up/down.

Button (real): Shown in Fig.7e, the IR agent is required to place the correct fruit in the bowl depending on the recipe. The IS policy is required to first turn the monitor on by pressing a button and then look at the screen to determine what fruit to use. Here the IR agent is a Tiago++ robot’s right hand that uses the images from its head camera along with its proprioceptives as observations and outputs a continuous delta action in the Cartesian space to control the hand. The IS policy observes the scene from Tiago’s head camera as well (see Appendix A.6) and in addition to controlling the head movement, also controls Tiago’s left arm. The IS policy has a discrete action space that allows it to pan and tilt the camera, along with controlling the second robot arm to press at various positions on the table.

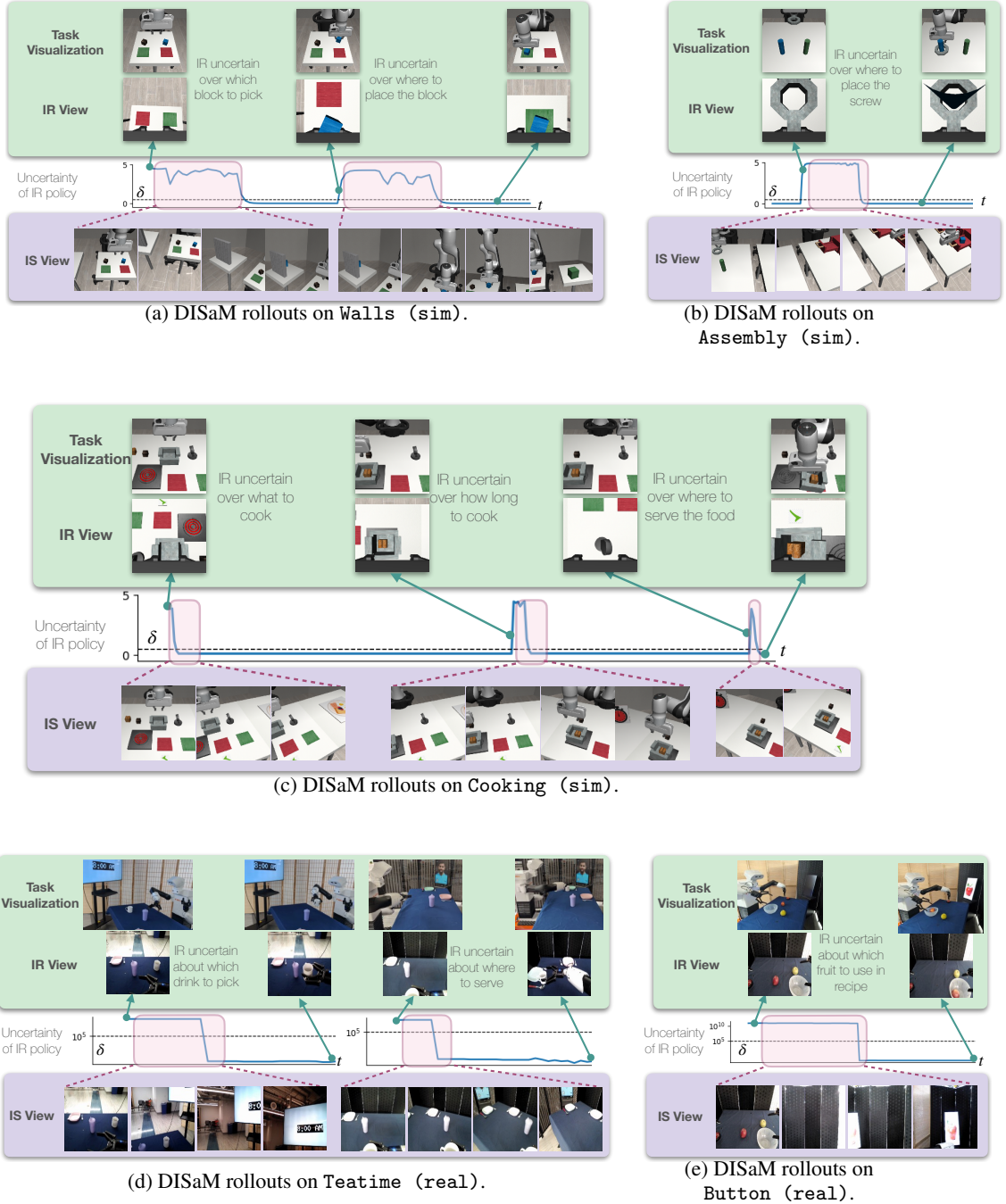


Figure 7: Here we demonstrate the rollouts of DISaM on the 5 tasks that we study. We show the uncertainty during the rollouts and 1) Highlight IR observations that have high uncertainty, and 2) Sequence of IS observations before finding information that reduce uncertainty.