

Figure 8: **Haptics-based Curiosity (HaC) Overview** *Top*: An agent perceives a scene visually and anticipates the force/torque (FT) sensation of interacting with an object. *Bottom*: The object interaction leads to an unexpected FT sensation, which gives a positive reward to the policy, leading to an exploration policy that is seeking interactions that are haptically surprising. The agent’s experiences gained in this way are later relabeled to become task-specific.

7 Supplementary Material

In this Supplementary Material section, we provide additional details concerning various elements which could not be elaborated on in the main paper. We begin with a detailed description of the proposed MiniTouch benchmark. We outline the action space, state space, and reward structure for each of the tasks. This is followed by a closer look into the various aspects of the haptics-control module including architectures, experimental procedures, hyper-parameters, and additional results. We include an ablation experiment at the end that investigates the usefulness larger exploration phase while solving each of the tasks.

7.1 MiniTouch Tasks

This section describes the cross-modal benchmark of simulated manipulation tasks, which we make use of and release as part of the paper. Unlike these prior simulation benchmarks, we particularly focus on providing a platform where one can use cross-modal information to solve diverse manipulation tasks. Existing benchmarks Yu et al. [55] do not include touch modality. An overview of the tasks in MiniTouch is outlined in Figure 2. These tasks are built off Pybullet [54] physics engine and contain different scene setups for each of the four tasks. The details of each of these tasks are further expanded. All the four tasks are compiled together as a “MiniTouch” benchmark suitable for evaluating interaction-based algorithms.

Playing: This environment is intended as a toy task to evaluate interaction frequency and does not feature any reward beyond interaction count. A cube is placed in a random position on a table at each episode. The agent needs to localize and interact with the cube.

Pushing: In this task, the agent needs to push an object placed randomly on a table to a target (visually indicated as a gray cube). The object position is sampled uniformly in polar coordinates around the target object (i.e. angle 0 to 360 degrees, distance 10 to 20 centimeters). The end effector’s start position is sampled in the same way as the target position. In addition, the orientation of the gripper is fixed to be perpendicular to the ground all the time. The robot agent succeeds and receives a reward of +25 if the distance between the cube and the target object is less than 7 centimeters. A new episode starts if the agent succeeds. The environment also restarts if the cube is placed or pushed beyond a predefined bounding box comprising of acceptable positions on the table (i.e. positions that can be reached by the robot hand).

Opening: A cabinet with a door is randomly placed in reach of the agent. The goal is to find the door handle and open the door. The gripper orientation is fixed to point its fingers towards the door, parallel to the ground. For this task, the fingers are discretized to be open or closed. In addition, a fifth element is added to the action vector to control the yaw (relative rotation of the end-effector) to be able to approach the door. The robot succeeds and receives a reward of +25 when the angle of the door opening reaches thirty degrees or higher. Similar to the pushing task, a new episode starts if the agent reaches the goal.

Pick-up: In this environment, the agent needs to grasp and lift a randomly placed object. The agent’s goal is to lift the object 5cm above the table. The agent receives a reward of +25 upon success. The object is placed uniformly randomly on a table. Similar to the Opening task, the end effector opening/closing is discretized, meaning when its internal continuous variable is below a threshold, the gripper closes, otherwise it remains open.

All of the tasks are implemented in the Pybullet physics engine [54], which is a free and open-source library that enables fast simulation.

7.2 Code

7.2.1 *MiniTouch* Library:

The task environment used in the experiments is packaged and released as a python library¹ that can be easily plugged into the training code. Setup instructions, code, and other details can be found in the README file included in the repository.

7.2.2 HaC and baselines:

We used the following open-source implementations for the baselines. We were able to reproduce the results from their papers before attempting to use them as baselines for our model:

ICM

<https://github.com/openai/large-scale-curiosity>

Disagreement:

<https://github.com/danijar/dreamerv2>

<https://github.com/pathak22/exploration-by-disagreement>

RND:

<https://github.com/openai/random-network-distillation>

7.3 Experimental Details

State space: The input states are a combination of visual and touch vector input. The visual input is a grayscale rendering of the scene with dimension 84×84 , pre-processed similarly to Mnih et al. [59]. Image observations are captured from a static camera overlooking each scene. The touch vector input is composed of the 3-dimensional end-effector position, 2-dimensional finger position, ranging from 0 to 1, each denoting how far apart each finger is, and the 6-dimensional force/torque values. In total, the touch vector is 11-dimensional, $S \in \mathbb{R}^{11}$.

Action space: Actions are expressed as 4-dimensional continuous vectors. The first three elements describe the desired offset for the end effector at the next timestep based on its current position. The last dimension controls the relative desired distance between the two fingers at the next timestep.

Training We use SAC [51] as the optimizer for our agent. Our training is composed of two phases as described in the main paper. (i)Exploration phase, (ii)Downstream task phase. In the exploratory phase (curiosity part) agent is trained using our intrinsic reward alone. In the task phase, network weights are seeded from the ones in the exploratory phase. We also retain the replay buffer from the exploratory phase in the downstream task phase. Duration of the exploration phase can be adjusted in the code using a hyper-parameter *stop-curiosity*. We have two hyper-parameters that change between the two phases, (i) α and (ii) the learning rate of the SAC algorithm. Details of hyper-parameters used for our experiments are outlined in Table 2 and Table 3.

¹<https://github.com/ElementAI/MiniTouch>

SAC pretraining and training hyperparameters	
Parameter type	Value
optimizer	Adam
Visual network	Table 3
learning rate	3.10^{-5}
number of samples per minibatch	128
reward scale	100
replay buffer size	10^6
number of hidden units per layer	128
number of hidden layers (all networks)	2
activation	LeakyReLU
discount factor	0.99

Table 2: SAC parameters during pretraining and training.

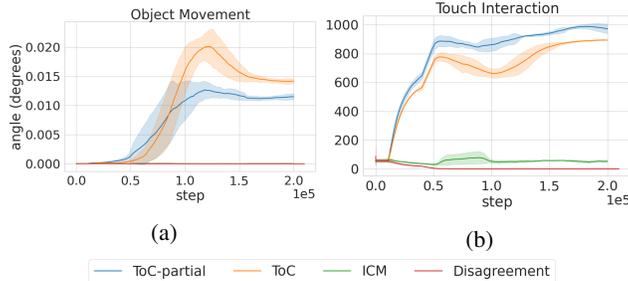


Figure 9: **Object Interaction for Opening task.** (a) shows the average variance in door angle across the entire episode (note that the absolute variance is low but corresponds to successful door openings towards the end of training for HaC) and in (b), we count the number of touch interactions in the same task.

Encoder network				
Layer	Number of outputs	Kernel size	Stride	Activation function
Input x	$84 * 84 * 1$			
Convolution	$20 * 20 * 32$	$8 * 8$	4	LeakyReLU
Convolution	$8 * 8 * 64$	$4 * 4$	2	LeakyReLU
Convolution	$4 * 4 * 124$	$3 * 3$	1	LeakyReLU
Convolution	$2 * 2 * 256$	$2 * 2$	1	LeakyReLU
Fully-connected	256	1		LeakyReLU

Table 3: Visual network.

7.4 Object Interaction

As touched up in the experiments section of the main paper, Figure 9 depicts the touch interaction and door movement metrics for the Opening task. We make a similar observation to the result showed in Figure 3 for the Playing task. A higher touch-interaction need not indicate better object-movement. Agent can resort to constantly engaging with the object in a passive manner. HaC collects rich interaction data during the exploratory phase and helps the agent in terms of sample efficiency while solving the downstream tasks.

7.5 Additional Ablations

Does longer exploration help? We observe that having a longer exploratory phase of training with intrinsic reward alone usually benefits the overall performance. We can observe from Figure 4 that HaC attains decent success in the exploratory phase without any external reward on most of the tasks. This is because it encourages better associations and a larger collection of interesting configurations in the replay. The effect of the exploratory step is further studied and the results on

Metric	Pushing			Open Door			Pick-up			Playing		
	HaC	RND+ <i>touch</i>	RND	HaC	RND+ <i>touch</i>	RND	HaC	HaC-RND	RND	HaC	RND+ <i>touch</i>	RND
Exploration \uparrow	0.403	0.181	0.077	0.669	0.564	0.380	0.063	0.013	0.005	-	-	-
Success \uparrow	0.733	0.710	0.582	0.983	0.921	0.875	0.891	0.593	0.450	-	-	-
Episode steps \downarrow	57.84	55.20	97.51	23.34	23.54	37.92	30.54	52.29	89.88	-	-	-
Touch-interaction \uparrow	247.79	227.75	206.02	600.1	411.43	194.22	980.7	894.5	725.1	388.15	312.0	124.5

Table 4: **Random Network Distillation (RND) with touch** This table compares the mean evaluations for HaC and RND+*touch* on all the four tasks emphasizing the importance of the cross-modal association(see text). We omit measuring success and episode steps for playing task since success there is equivalent to object-interaction and has no explicit goal.

all of the downstream tasks with different duration of exploration are compiled in Figure 10. We depict the episode convergence steps and success rate to visualize the trend as expected.

Touch in Random Network Distillation We created an additional baseline RND+*touch* similar to touch in future prediction (ICM+*touch*) baseline. In this setting, the input to the random target network and predictor network (that predicts the outcome of target network) consists of both touch and visual feature vector concatenated. The baseline beats the vision only version by a margin for both Pushing and picking tasks as shown in the Table. 4, however for the Open-door tasks the improvement is not that significant. Although RND+*touch* is comparable with HaC for the Pushing task on the success rate, HaC has better sample efficiency as measured by episode steps. We believe this is because object movement is crucial for the pushing task compared to interaction, and prediction based techniques such as RND could be helpful in such settings.

Comparisons with Generalization to Novel shapes: It is desirable for an agent to be able to handle diverse shapes in order to be robust across arbitrary manipulation settings. We study this using an environment in which an object is sampled from a thousand procedurally generated objects. The objects are dissimilar with respect to shape and mass but are sampled from the same generative distribution. Out of 1000 different objects, 800 of them are used in the training phase, and we evaluate the agent’s effectiveness on the remaining 200 unseen object shapes.

Although generalization to novel shapes is not the problem setting our method focuses on. We perform this proof of concept ablations to investigate that HaC parameters in the exploratory phase are not just reusable across different tasks with similar shapes (Eg: Playing, Pushing, and Picking), but also can help generalize when applied to distinctive shapes sampled from a similar distribution. Figure 11 shows touch interaction and object movement evaluations for a single object exploration task. We compare HaC with ICM and Disagreement baselines. . The results validate that our model generalizes to unseen object configurations.

7.6 Real world use cases:

It is possible to define safety boundaries for a robot arm (e.g. don’t hit the table, don’t move outside the arena boundaries) and have a robot arm autonomously and task-independently collect data for phase 1 of our method. In addition to this, we would like to point out that (a) in the pushing and pick-up tasks, only 5-10k steps are required for decent performance on the downstream task and (b) what we call a “step” is however long it takes the inverse kinematic solver to move the arm to the new end effector pose (that was generated by the policy). If many of the generated poses cluster together, that dramatically reduces runtime of the curiosity phase making it more suitable to real-world applicability. While no real robot data have been used in the experiments, our method is data agnostic. We note that the physics engine can be replaced by real-world data as there is no bias introduced in process. The networks used for the curiosity prediction task can be easily adapted, if needed, to accommodate inputs from a real robotic platform. As a next step and when restrictions are eased, we plan to implement this method on a physical system and replicate our experiments.

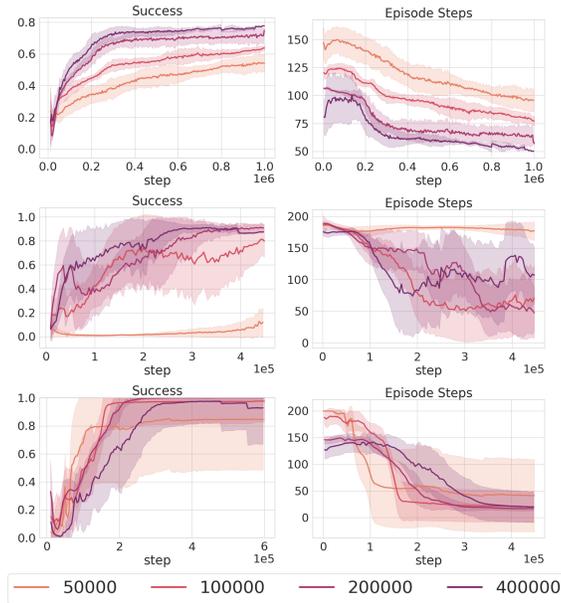


Figure 10: **Longer exploratory phase helps** Success and episode steps evaluations on different tasks for different lengths of *exploratory* phase. Darker shading indicates longer exploration.

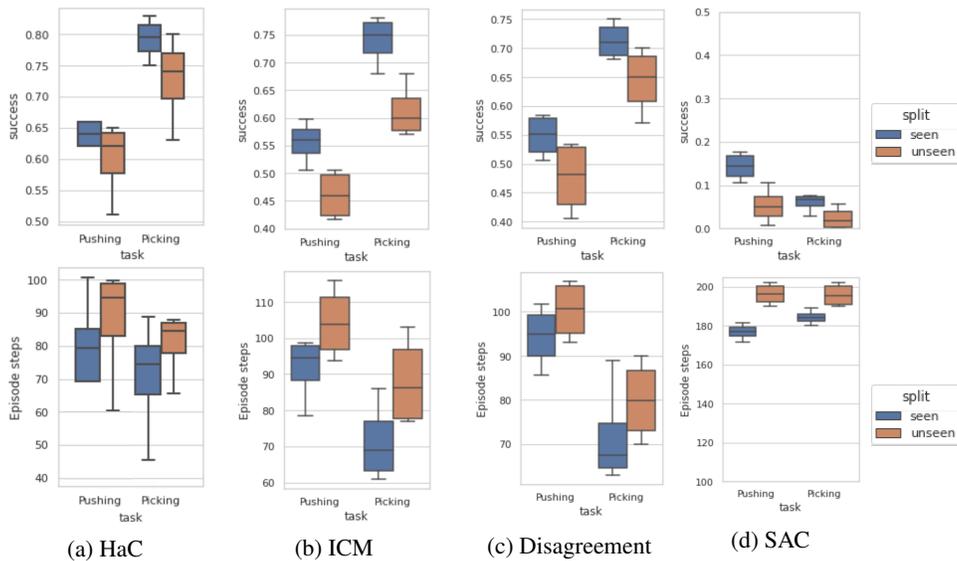


Figure 11: **Generalization to novel shapes baseline comparisons.** Each column outlines Success and Episode steps for HaC, ICM, Disagreement and SAC baselines respectively. Disagreement has better generalization and reduced variance compared to ICM on the unseen categories. HaC is comparable to Disagreement on generalization performance while displaying better success and Episode steps rates.