Active Perception for Tactile Sensing: A Task-Agnostic Attention-Based Approach

Tim Schneider[1,2], Cristiana de Farias[1], Roberto Calandra[3], Liming Chen[2], and Jan Peters[4]

[1]Department of Computer Science, TU Darmstadt, Germany.
[2]LIRIS, CNRS UMR5205, Ecole Centrale de Lyon, France.
[3]LASR Lab & CeTI, TU Dresden, Germany.
[4]DFKI, Hessian.AI, and Centre for Cognitive Science, TU Darmstadt, Germany.

Abstract

Humans make extensive use of haptic exploration to map and identify the properties of the objects that we touch. In robotics, active tactile perception has emerged as an important research domain that complements vision for tasks such as object classification, shape reconstruction, and manipulation. This work introduces `TAP` (Task-agnostic Active Perception) – a novel framework that leverages reinforcement learning (RL) and transformer-based architectures to address the challenges posed by partially observable environments. `TAP` integrates Soft Actor-Critic (`SAC`) and `CrossQ` algorithms within a unified optimization objective, jointly training a perception module and decision-making policy. By design, `TAP` is completely task-agnostic and can, in principle, generalize to any active perception problem. We evaluate `TAP` across diverse tasks, including toy examples and realistic applications involving haptic exploration of 3D models from the Tactile MNIST benchmark. Experiments demonstrate the efficacy of `TAP`, achieving high accuracies on the Tactile MNIST haptic digit recognition task and a tactile pose estimation task. These findings underscore the potential of `TAP` as a versatile and generalizable framework for advancing active tactile perception in robotics.

1 Introduction

Tactile sensing plays a fundamental role in perception and manipulation, spanning practical tasks such as object grasping and broader perceptual tasks, including material and shape recognition, pose estimation, and localization. While its significance for human manipulation has been well established [1, 2, 3], recent advances in tactile sensing hardware have not only made it more accessible but also increased its popularity in robotics. Notably, while tactile sensing provides rich, local information about contact interactions, it lacks the wider coverage of other sensing modalities (e.g., vision), thereby necessitating exploratory procedures to enable information gathering [4]. Because exhaustive tactile exploration is often prohibitively expensive, active perception emerges as a promising solution, leveraging tactile observations from the agent to better guide its interaction with the environment.

Recent active tactile perception methods have addressed shape estimation [5, 6], texture recognition [7], object classification [8, 9], and grasping [10], often through specialized exploration strategies. Particularly, Fleer et al. [8] developed the Haptic Attention Model (`HAM`), which builds upon the recurrent model of visual attention proposed by Mnih et al. [11]. `HAM` jointly optimizes perception and action using REINFORCE, enabling the classification of objects through haptic exploration.

However, while these approaches showcase significant progress, they are often tuned to their specific tasks or sensors, which limits their versatility. Humans, conversely, demonstrate versatile tactile capabilities, performing diverse tasks without specialized retraining. Achieving similar generality in robotics requires flexible strategies for sequential decision-making that go beyond task-specific objectives. Unlike supervised methods based on static observations, tactile sensing demands dynamic exploration strategies that balance exploration and exploitation. The challenge is compounded by the fact that touch involves physical interaction, which can actively alter the environment's state. Nonetheless, many prior works assume a static setting where objects remain stationary during exploration [12, 5, 13, 10], simplifying the problem.

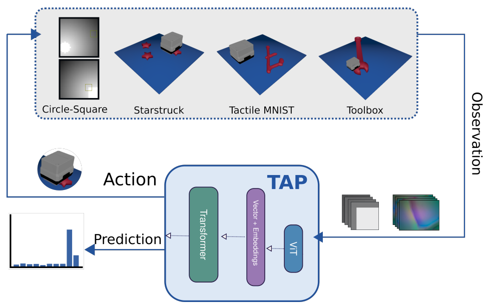

Figure 1: Our method ***T**ask-agnostic **A**ctive **P**erception* (`TAP`) aims to infer properties, such as object classes, of its environment based on limited per-step information. To do so, it jointly optimizes an action policy to gather information and a prediction model for inference. Both the action policy and the prediction model use a shared transformer-based backbone to process sequences of inputs. Illustrated on the top are four benchmark tasks we use to evaluate `TAP`.

In this manner, partially observable strategies such as the partially observable Markov decision process (POMDP) framework provide a formal foundation for addressing these challenges. POMDPs enable modeling sequential decision-making in environments where the agent's observations are inherently partial or noisy. For efficient exploration, reinforcement learning (RL) naturally fits active perception, enabling task-adaptive exploration policies through reward maximization while balancing exploration and exploitation in uncertain and dynamic settings. In this context, Shahidzadeh et al. [14] introduced an RL-based method for active tactile exploration, specifically for the task of tactile shape reconstruction.

In this work, we introduce Task-agnostic Active Perception (`TAP`), combining RL and supervised learning in a general framework that requires only a differentiable loss and a POMDP environment. Our method jointly optimizes a policy and perception module using a shared transformer backbone (see Fig. 1), accommodating diverse sensor inputs without task-specific adjustments. Here, we propose two variants of `TAP`, extending `SAC` [15] and `CrossQ` [16]. To validate our method, we evaluate it on four different benchmarks, ranging from classification to counting and localization tasks. Our experiments confirm RL's effectiveness for tactile exploration, though improving sample efficiency remains challenging. In summary, our main contributions are:

- A task-agnostic framework that jointly optimizes an RL policy and perception module using a shared transformer backbone. This formulation allows both adaptability across tasks and robustness to dynamic environments where objects may move or change during interaction.
- Two method variants, based on SAC and CrossQ, are designed for a range of active tactile exploration tasks.
- Empirical validation across multiple tactile perception benchmarks, demonstrating the method's generality and effectiveness.

2 Related Work

We briefly review prior work on tactile perception, vision-based tactile sensing, and active tactile exploration. For comprehensive surveys on tactile manipulation, we refer readers to [18, 19].

Tactile Perception: Tactile sensing allows robots to infer object geometry, texture, and materials through physical contact, complementing or substituting visual sensing, especially in occluded scenarios. Early tactile sensing systems primarily used simple binary contacts [19], whereas recent approaches employ vision-based tactile sensors [20, 21, 22]. These sensors provide high-resolution data useful in complex tasks, including shape reconstruction, texture recognition, and advanced manipulation, such as autonomous page turning [23], object reorientation [24, 25], and handling deformable objects [26]. Additionally, transformer architectures, known for capturing long-range dependencies in vision tasks, have recently been adapted for tactile perception due to their ability to

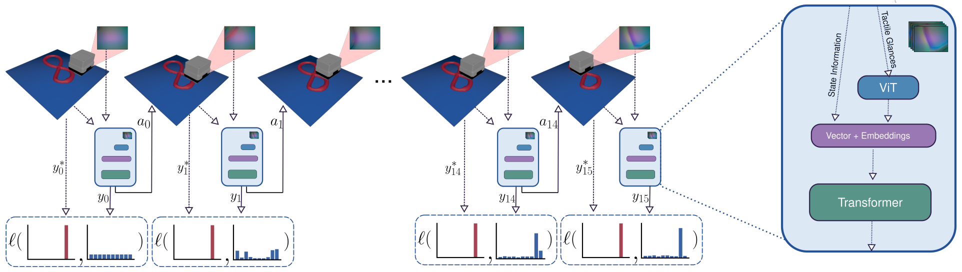

Figure 2: Overview of our Task-agnostic Active Perception (TAP) method. The agent aims to classify the 3D MNIST digit using touch alone. At each step, it receives a tactile reading and state information (e.g., sensor position). A Vision Transformer (ViT) [17] encodes the tactile input, which is concatenated with state data and processed as a sequence by a transformer. The model outputs a label prediction y_t, evaluated against the ground truth $\overset{*}{y}$ via a loss function ℓ, and an action a_t that controls the sensor's next movement.

integrate spatially detailed and temporal tactile signals [27, 28]. Following this trend, in this work, we adopt a transformer-based perception module to robustly process sequential tactile data.

Active Tactile Exploration: Active tactile exploration involves deliberately selecting contact locations and trajectories to optimize information gain during interaction. Previous approaches leverage Gaussian Processes [12, 5] and Bayesian optimization [13, 10, 7] to efficiently reconstruct shapes, discriminate textures, or identify grasp points. RL methods have also emerged for tactile exploration in high-dimensional state spaces, dynamically selecting discriminative contact regions to enhance perception [9]. Closely related to this work, [11] proposes the Recurrent Models of Visual Attention (RAM) for active vision tasks, where a selective attention mechanism is designed to optimize scene exploration. Extending this concept to tactile data, the Haptic Attention Model (`HAM`) [8] tailored RAM for tactile perception by optimizing selective attention. In contrast, we frame active tactile perception explicitly as a supervised learning task within a POMDP. To address sequential decision-making, we employ state-of-the-art RL algorithms such as Soft Actor-Critic (`SAC`) [15] and `CrossQ` [16]. This formulation enables our pipeline to adapt to a variety of tasks while remaining robust to dynamic environments where objects may move or change during interaction.

3 Active Perception

In this section, we formally derive our method for active perception. We propose two variants of our method, based on `SAC` [15] and `CrossQ` [16].

3.1 Problem Description

Formally, we define the problem of active perception as a special case of a Partially Observable Markov Decision Process (POMDP). Here, the environment is governed by unknown dynamics $p(\hat{h}_{t+1}|\hat{h}_t, \hat{a}_t)$, where $\hat{h}_t$ is the hidden environment state and $\hat{a}_t$ is the action taken at time t. The agent then makes observations through the distribution, $p(o_t|\hat{h}_t)$, where o_t is the observation. In the active perception scenario, the agent's objective is to learn a particular property of the environment, e.g., the class or pose of an object. We assume that the ground truth value $\overset{*}{y}_t$ of this property at time t is part of the hidden state $\hat{h}_t$ and thus not directly accessible to the agent. Hence, the hidden state decomposes into $\hat{h}_t = (h_t, \overset{*}{y}_t)$, where h_t is the remainder of the hidden state without the ground truth property value. Additionally, the agent's action space contains not only control actions a_t but also a current estimate y_t of the desired environment property. In other words, the action space decomposes into $\hat{a}_t = (a_t, y_t)$, meaning that the agent predicts the desired environment property at every step.

As typical for RL, the action a_t is a control signal, e.g., a desired finger movement, which is communicated to the agent's motor controllers. The resulting reward function now consists of two parts: a differentiable prediction loss ℓ and a regular RL reward r. That is, $\hat{r}(h_t, \overset{*}{y}_t, a_t, y_t) = r(h_t, a_t) - \ell(\overset{*}{y}_t, y_t)$. Here, the prediction loss, $\ell(\overset{*}{y}_t, y_t)$ could, for example, be a cross-entropy loss in the case of a classification task or the Euclidean distance in the case of a pose estimation task. The

RL reward, on the other hand, does not have to be differentiable or known to the agent and could be any function. In this work, we only use it to regularize the agent's actions a_t. In the following, to simplify the notation, we use $p\Big(\mathbf{h}, \overset{*}{\mathbf{y}}, \mathbf{o}, \mathbf{a}, \mathbf{y}\Big) = \pi_\theta(\mathbf{o} \,|\, \mathbf{h})\; p\Big(\mathbf{h}, \overset{*}{\mathbf{y}}, \mathbf{o}, \mathbf{a}\Big)$, $\pi(\mathbf{y} \,|\, \mathbf{o}) = \prod_{t=0}^{\infty} \pi(y_t \,|\, o_{0:t})$ and $p\Big(\mathbf{h}, \overset{*}{\mathbf{y}}, \mathbf{o}, \mathbf{a}\Big) = p\Big(h_0, \overset{*}{y}_0\Big) \prod_{t=0}^{\infty} p(o_t \,|\, h_t)\, \pi(a_t \,|\, o_{0:t})$ where $\mathbf{h} \coloneqq h_{0:\infty}$, $\overset{*}{\mathbf{y}} \coloneqq \overset{*}{y}_{0:\infty}$, and so on. Thus, the objective is now to find a policy $\pi(a_t \,|\, o_{0:t})$ for which the expected discounted return is maximized. That is, given the discount factor $\gamma \in [0, 1)$,

$$\max_{\pi} J(\pi) \coloneqq \mathop{\mathbb{E}}_{p\left(\mathbf{h}, \overset{*}{\mathbf{y}}, \mathbf{o}, \mathbf{a}, \mathbf{y}\right)} \left[\sum_{t=0}^{\infty} \gamma^t \hat{r}(h_t, \overset{*}{y}_t, a_t, y_t) \right] \tag{1}$$

3.2 Optimizing the Active Perception Objective

Since the agent's predictions y_t do not influence future states, by defining $\ell_\pi(\overset{*}{y}_t, o_{0:t}) \coloneqq \mathbb{E}_{\pi(y_t \,|\, o_{0:t})}\left[\ell(\overset{*}{y}_t, y_t)\right]$ we can rewrite (1) as

$$J(\pi) = \mathop{\mathbb{E}}_{p\left(\mathbf{h}, \overset{*}{\mathbf{y}}, \mathbf{o}, \mathbf{a}\right)} \left[\sum_{t=0}^{\infty} \gamma^t \left(r(h_t, a_t) - \ell_\pi(\overset{*}{y}_t, o_{0:t}) \right) \right]. \tag{2}$$

In this work, we assume that the policy π is parameterized by parameters $\theta \in \mathbb{R}^M$, which allows us to compute a gradient of (2) and optimize the problem with gradient descent algorithms. Computing the gradient of $J(\pi_\theta)$ now yields

$$\frac{\partial}{\partial \theta} J(\pi_\theta) = \underbrace{\mathop{\mathbb{E}}_{p_\theta\left(\mathbf{h}, \overset{*}{\mathbf{y}}, \mathbf{o}, \mathbf{a}\right)} \left[\frac{\partial}{\partial \theta} \ln \pi_\theta(\mathbf{a} \,|\, \mathbf{o}) \sum_{t=0}^{\infty} \gamma^t \hat{r}(h_t, \overset{*}{y}_t, a_t, y_t) \right]}_{\text{policy gradient}} - \underbrace{\mathop{\mathbb{E}}_{p_\theta\left(\overset{*}{\mathbf{y}}, \mathbf{o}\right)} \left[\sum_{t=0}^{\infty} \gamma^t \frac{\partial}{\partial \theta} \ell_{\pi_\theta}(\overset{*}{y}_t, o_{0:t}) \right]}_{\text{prediction loss gradient}}. \tag{3}$$

As shown in (3), the gradient of the objective function $J(\pi_\theta)$ decomposes into a policy gradient and a negative supervised prediction loss gradient.

3.3 Deriving `TAP-SAC` and `TAP-CrossQ`

We use RL-based techniques to estimate the policy gradient in Eq. (3), focusing on two actor-critic methods: `SAC` [15] and `CrossQ` [16]. `SAC` is an off-policy RL algorithm that jointly learns a policy and two Q-networks. The Q-networks are optimized to minimize the Bellman residual for the policy, and the policy is optimized to maximize the smaller of the two values predicted by the Q-networks. To stabilize the training of the Q-networks, `SAC` deploys target networks, which slowly track the actual Q-networks. `CrossQ` is similar to `SAC` but drops the target networks and instead uses BatchRenorm [29] layers in the Q-network to stabilize their training. To use `SAC` and `CrossQ` in the active perception setting, we adjust three components:

1. The active perception setting is partially observed. Hence, instead of a state s_t, the policy and the Q-networks receive a trajectory of past observations $o_{0:t}$.

2. During the training of the Q-networks, `SAC` and `CrossQ` sample transitions and rewards from the replay buffer to compute the Bellman residual. The presence of the prediction loss ℓ_{π_θ} requires us to dynamically recompute the total reward when evaluating the Bellman residual, yielding

$$\mathcal{L}_{\text{critic}} = \mathop{\mathbb{E}}_{\mathcal{D}} \left[\frac{1}{2} \left(Q_\theta(o_{0:t}, a_t) - \left(r_t - \ell_{\pi_\theta}(\overset{*}{y}_t, o_{0:t}) + \gamma \mathop{\mathbb{E}}_{\pi_\theta}[Q_{\bar{\theta}}(o_{0:t+1}, a_{t+1})] \right) \right)^2 \right].$$

3. During the policy update, the policy gradient is augmented by the prediction loss gradient.

For the remainder of this paper, we will refer to the `SAC` variant as `TAP-SAC` and to the `CrossQ` variant as `TAP-CrossQ`.

3.4 Input Processing

We assume that the sequence of past observations $o_{0:t}$ consists of images and scalar data. To efficiently process this data in the policy and Q-networks, we use an architecture similar to the Video-Vision-Transformer (ViViT) [30] architecture. First, a Vision Transformer (ViT) [17] is used to generate embeddings for each tactile image. These embeddings are then concatenated with the scalar inputs and processed by a transformer to generate an embedding m_t for each time step. We empirically found that sharing these embeddings across Q-networks $Q_\theta(o_{0:t}, a_t)$, action policy $\pi_\theta(a_t \mid o_{0:t})$ and prediction policy $\pi_\theta(y_t \mid o_{0:t})$ yields better results than training individual representations for each of these components. A full overview of `TAP` is shown in Fig. 2.

4 Experiments

To evaluate `TAP`, we conduct experiments on four active perception tasks: *Circle-Square*, *Tactile MNIST*, *Starstruck*, and *Toolbox* [31] (see Fig. 3). In each task, the agent must actively gather information, jointly learning both a policy and a perception model.

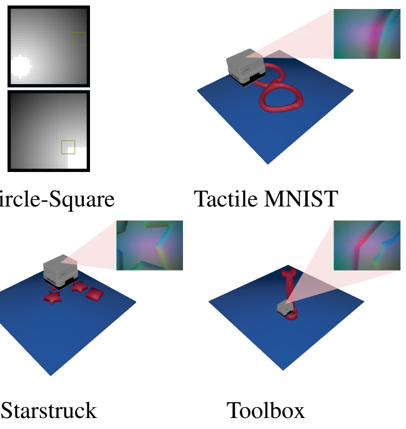

Figure 3: Active perception benchmarks on which we evaluate our method. Tactile MNIST, Starstruck, and Toolbox are tactile perception tasks from the *Tactile MNIST benchmark* [31]. In each environment, the agent must decide how to gather information with its sensor. Circle-Square and Tactile MNIST are classification tasks, where the agent must decide on a class label. Starstruck is a searching and counting task, where the agent must determine the number of stars in the environment. Toolbox is a pose estimation task, where the agent must determine the 2D pose of the given object. All tasks require the agent to gather information actively and are not accurately solvable via random exploration.

4.1 The Circle-Square Task

The Circle-Square task is an environment for evaluating active perception in a low-dimensional space. Here, the agent is presented with a 28×28 grayscale image containing either a white circle or square placed randomly in the field (Fig. 3-top-left). Its goal is to identify the correct object class, but it can only observe a 5×5 glimpse at a time and must explore the image over time. Each episode allows the agent to take up to 16 steps. A color gradient offers directional guidance, but the agent starts without information about the object's location. It selects a continuous movement action $a_t \in [-1, 1]^2$ per step to reposition its glimpse, encouraging learned search strategies over random behavior. Since the agent's glance in this task is small, we do not use our vision encoder but treat the inputs as a 25-dimensional vector. By not using the vision encoder, `TAP` becomes more directly comparable to `HAM` [8], which also processes a flat representation of the input image.

In addition to the `HAM` baseline, we compare our method to a random baseline agent, `TAP-RND`. `TAP-RND` shares the same configuration as `TAP-SAC` but does not train an action policy; instead, it selects all actions randomly. Despite this, `TAP-RND` still trains the perception module, allowing it to learn to process data without having control over the location of its haptic glances. We use each agent's average and final class prediction accuracies as our evaluation metrics. Here, the *average class prediction accuracy* takes the agent's predictions in all steps of an episode into account and computes the average over those. The *final class prediction accuracy*, on the other hand, takes only the prediction in the final step of each episode into account. We ran this experiment on 5 seeds per method, training all models from scratch each time.

As shown in Fig. 4, our agents `TAP-SAC` and `TAP-CrossQ` learn to complete the task with similar final prediction accuracies of 97% and 96%, respectively. The random agent, `TAP-RND`, achieves a 68% accuracy, highlighting the need for active perception in the Circle-Square task. Despite extensive tuning of the learning rate and β parameter with `HEBO` and training for 10M steps, we could not find a configuration for which `HAM` reached a prediction accuracy beyond random guessing. To understand

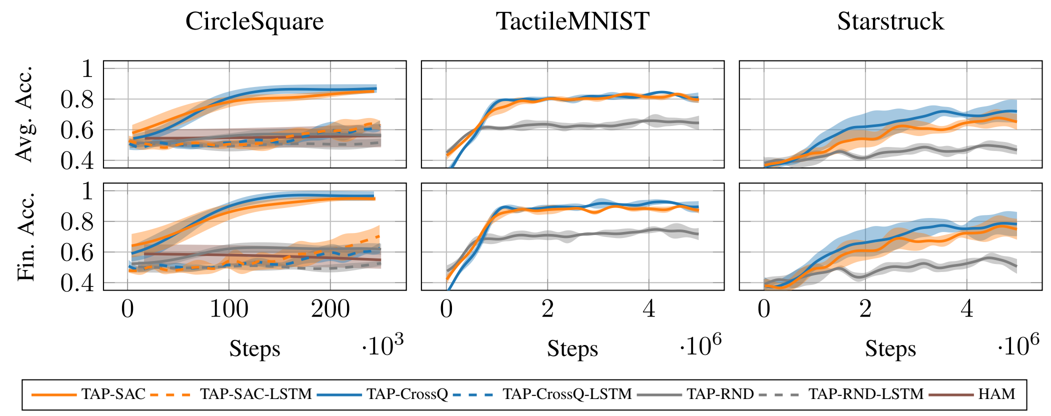

Figure 4: Average and final prediction accuracies for our methods `TAP-SAC` and `TAP-CrossQ`, `HAM` [8], and a random baseline `TAP-RND` across various classification tasks. Variants `TAP-SAC-LSTM`, `TAP-CrossQ-LSTM`, and `TAP-RND-LSTM` use the LSTM model from `HAM` in place of our transformer. All methods were trained with 5 seeds. Shaded areas represent one standard deviation. Metrics are computed on evaluation tasks with unseen objects, except for Circle-Square, which has only two. For Starstruck, a correct classification requires predicting the exact number of stars.

what leads to this performance gap, we ran an ablation, using `HAM`'s LSTM model in combination with `TAP`'s off-policy optimization method. The results of this ablation are shown in Fig. 4, where we call these methods `TAP-SAC-LSTM`, `TAP-CrossQ-LSTM`, and `TAP-RND-LSTM`. While the LSTM-based approaches still fall short of the Transformer-based approaches, they achieve higher accuracies than `HAM`, suggesting that `HAM`'s poor performance in this task is at least partly to blame on its on-policy optimization method. For full implementation details, including an ablation with longer training times, where we compare `HAM` with a PPO baseline, see Appendix A.1.

4.2 The Tactile MNIST Benchmark

While the Circle-Square environment already presents a non-trivial active perception problem for task-agnostic agents, it remains relatively simple: the input space contains just 25 pixels, there are only two object classes, with a color gradient providing a search direction. In contrast, (visual-) tactile perception introduces greater complexity. First, the input is a high-dimensional image requiring encoding into a latent space. Second, real-world classification tasks often involve many classes with diverse instances. For example, a robot sorting waste into plastics, glass, and metal must handle objects of various shapes and textures that belong to the same class. Finally, tactile exploration often lacks directional cues, as retrieving an object from a cluttered bag requires systematic search strategies. To investigate this, we evaluate our methods on the *Tactile MNIST benchmark* introduced by Schneider et al. [31]. In this task, an agent uses a GelSight Mini sensor to explore a randomly placed and oriented 3D MNIST digit without prior location knowledge. The goal is to classify the digit within a fixed time budget (see Fig. 3-top-right for a visualization).

As shown in Fig. 4, both `TAP-SAC` and `TAP-CrossQ` reach similar high final prediction accuracies of 87% and 89% on the evaluation task. The `TAP-RND` baseline, however, eventually stagnates at an accuracy of around 74%, highlighting the importance of action selection on this task. The average class prediction accuracy, as shown in Fig. 4(first column), presents a similar trend, with `TAP-SAC` achieving 80% and `TAP-CrossQ` 81%. An insight into the agent's performance is given by Fig. 5, which shows how the accuracy and correct label probability of the trained agents over the course of an episode, averaged over multiple episodes. This figure shows that the active agents gather information much quicker and are much more certain about the class label than the random agent. For full benchmark setup, training details, and sensor configuration, see Appendix A.2.

4.3 Starstruck

The *Starstruck* task (see Fig. 3-bottom-left) builds on the Tactile MNIST simulation framework but introduces a distinct challenge. Rather than classifying an object, the agent must count how many stars are placed among randomly arranged geometric shapes (stars, squares, and circles). Since all stars share the same 3D model, this task removes intra-class variation and focuses purely on exploration, allowing us to isolate systematic search behavior. Here, at the start of each episode, between one and three stars are placed randomly among distractor shapes. The agent's goal is to determine the exact number of stars on the plate using the GelSight Mini sensor. As in Tactile MNIST, the agent moves freely in 2D space under the same motion constraints, but we extend the episode budget to 32 steps to accommodate the larger search space. We frame this task as a 3-way classification problem where the correct prediction must exactly match the star count. We evaluate `TAP-SAC`, `TAP-CrossQ`, and the random baseline `TAP-RND`, using the same hyperparameters as in the Tactile MNIST task to ensure comparability.

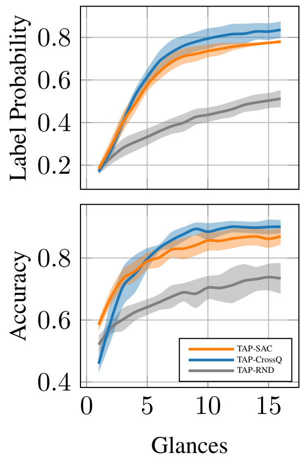

Figure 5: Exploration efficiency of final policies on the Tactile MNIST benchmark. Shown are the predicted probability of the correct label (top) and accuracy (bottom) after N tactile glances.

The results, summarized in Fig. 4, indicate that `TAP-CrossQ` reaches a final prediction accuracy of 78%, with an average prediction accuracy of 72%. `TAP-SAC` achieves a similar final accuracy of 75% and a slightly lower average accuracy of 65%, suggesting that both approaches are well-suited for this task. In contrast, the `TAP-RND` baseline peaks at a 56% final accuracy and 50% average accuracy, underscoring the necessity of active perception in solving this problem. The slower convergence compared to Tactile MNIST suggests that the Starstruck task is more challenging to solve for our methods. However, this could, in part, be due to the increased episode length of 32 vs 16 in Tactile MNIST, which effectively reduces the number of episodes seen during training to half in comparison to Tactile MNIST. For implementation details and hyperparameter settings, see Appendix A.3.

4.4 Toolbox

In the *Toolbox* task, the agent has to find a wrench on a platform by touch alone. Unlike Tactile MNIST and Starstruck, Toolbox is a regression task, where the agent has to predict the 2D position and 1D orientation of the wrench in the workspace. As such, the task consists of two problems: the agent must find the wrench in the workspace and determine its position and orientation. Crucially, as can be seen in Fig. 3-bottom-right, most parts of the wrench are ambiguous in location when touched, and the information of multiple touches has to be combined to make an accurate prediction. For example, when touching the handle, the agent may extract information about the lateral position of the wrench, but does not yet know where it is currently touching the handle longitudinally, and whether the open end is left or right. Hence, to disambiguate the wrench pose, a strong exploration strategy must include finding one of the ends of the wrench. Similar to the previous two tasks, the agent moves a GelSight Mini sensor freely in 2D space, constrained by the platform boundaries. Since the platform for this task is larger than that for the other tasks, we allow 64 steps for exploration before the episode is terminated.

The results, shown in Fig. 6, show that `TAP-CrossQ` reaches a final accuracy of 1.9cm and 13° on average, while `TAP-SAC` and the random baseline `TAP-RND` stagnate at much lower accuracies. The low performance of the `TAP-RND` again highlights the importance of active perception for this task. It is important to note that we again did not tune any of these methods on this task and instead relied on hyperparameters optimized for Tactile MNIST. While tuning `TAP-SAC` on this task can lead to stronger performance, these results, together with the Starstruck results in Fig. 4, indicate that `TAP-CrossQ` might be more robust w.r.t the choice of hyperparameters. However, more experiments are needed to answer this question definitively. For training details, evaluation protocol, and hyperparameter settings, see Appendix A.4.

5 Discussion

The results in Section 4 show that our task-agnostic, RL-based method successfully learns exploration strategies across diverse active perception tasks. We evaluated TAP on a toy Circle-Square classification task and three simulated tactile benchmarks: Tactile MNIST (digit classification), Starstruck (object counting), and Toolbox (object pose estimation). In all three tactile benchmarks, the agent must learn an image encoder and refine its exploration policy jointly. Notably, TAP-CrossQ retained high performance when switching tasks without hyperparameter tuning, highlighting its robustness.

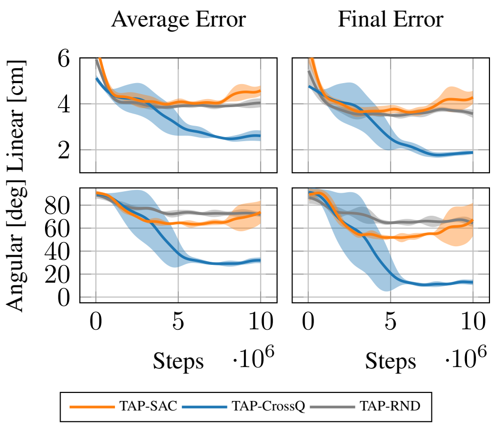

Figure 6: Average and final prediction accuracies over five runs for our methods TAP-SAC and TAP-CrossQ, as well as the baseline TAP-RND on the Toolbox task. Each method was trained on 5 seeds for 10M steps.

The poor performance of the TAP-RND baseline across all tasks confirms the necessity of structured exploration, while HAM [8] failed to learn an effective strategy on Circle-Square, defaulting to predicting the mean class despite extensive tuning using HEBO. This points to a fundamental limitation of HAM in this setting. Key differences between TAP and HAM explain this gap: first, TAP uses a transformer-based policy network, which our ablation shows outperforms LSTM-based variants, likely due to better modeling of long-term dependencies. Second, HAM relies on on-policy RL, which discards samples after a single update, limiting learning efficiency. In contrast, TAP employs off-policy methods (TAP-SAC and TAP-CrossQ), enabling sample reuse – a critical factor in active perception, where supervised learning benefits from multiple passes over the same data. Both TAP-SAC and TAP-CrossQ perform comparably on environments they have been tuned on, but TAP-CrossQ has proven more robust to new environments without hyperparameter tuning and offers a clear computational advantage. By avoiding target network updates, it requires roughly half the transformer forward passes during training, leading to a 53% reduction in training time on average, without sacrificing performance.

In summary, transformer-based policies and off-policy RL significantly improve active perception learning. The strong cross-task performance, especially of TAP-CrossQ without retuning, demonstrates its potential as a robust, general framework for active tactile perception.

6 Conclusion

We introduced TAP (Task-agnostic Active Perception), a framework combining reinforcement learning and transformer-based models for active tactile perception. We evaluated it on four benchmarks where TAP consistently outperforms a random baseline, underscoring the value of active perception. Notably, the current state-of-the-art method – HAM – cannot solve any of these tasks. On the Circle-Square task, HAM resorted to always predicting a 50/50 probability for both labels, even after long training, while the other three tasks require an image encoder, which HAM lacks. TAP, on the other hand, achieves high performance across all tasks, with the exception of TAP-SAC on the Toolbox task, where it was not tuned on. Ablation results on Circle-Square additionally show that our transformer-based architecture outperforms HAM's LSTM, even when paired with our actor-critic training. Future work will focus on applying sim-to-real transfer for deployment, extending TAP to real-world applications such as in-hand pose estimation and texture recognition, and exploring multi-modal integration with vision and touch.

7 Limitations

While TAP demonstrates strong performance across various active perception tasks, it also has certain limitations. One of the primary drawbacks is its reliance on large amounts of training data, requiring up to 5M steps for the tactile perception tasks. This high data requirement arises from the combination of a transformer-based architecture and RL-based policy optimization. While this approach enhances the generality of TAP, allowing it to adapt to different tasks without hyperparameter tuning, it comes at the cost of sample efficiency. A promising avenue to solve this issue is leveraging pre-trained transformer models, which could improve sample efficiency by providing useful feature representations. Furthermore, recent advancements in sample-efficient reinforcement learning [32, 33] offer potential alternatives for improving the practicality of TAP in real-world applications.

Another limitation of the proposed work, and not necessarily from the proposed method, that is TAP, is the lack of real-world experimentation. This is an avenue that we aim to investigate in future work, integrating state-of-the-art methods for sim-to-real transfer known in RL. Domain randomization and adaptation offer potential pathways to mitigate this limitation by enabling policies trained in simulation to generalize to real-world environments. Additionally, approaches such as CycleGAN [34] can be used to enhance the realism of simulated tactile images, improving the alignment between simulated and real-world data to facilitate transfer.

Another similar limitation and future direction is to explore a more diverse and practical set of tasks. Applications such as object pose estimation, shape reconstruction, or material property inference remain unexplored and could pose additional challenges to our methodology. Moreover, our current experiments use a single tactile sensor, but in principle, the TAP model architecture supports multi-fingered robotic hands and multi-modal perception (e.g., combining vision and touch). However, the practical scalability of TAP to those applications remains an open question, as the increased action and observation space complexity may introduce additional challenges in training efficiency, policy learning stability, and computational demands. Future work will explore these extensions by evaluating TAP on multi-fingered robotic systems and integrating complementary sensing modalities to enhance active perception capabilities.

References

[1] Roland S Johansson. Sensory input and control of grip. In *Novartis Foundation Symposium*, pages 45–58. Wiley Online Library, 1998.

[2] Roland S Johansson. Sensory control of dexterous manipulation in humans. In *Hand and brain*, pages 381–414. Elsevier, 1996.

[3] Jenmalm, Per and Dahlstedt, Seth and Johansson, Roland S. Visual and tactile information about object-curvature control fingertip forces and grasp kinematics in human dexterous manipulation. *Journal Neurophysiology*, 84(6):2984–2997, 2000.

[4] Tony J. Prescott, Mathew E. Diamond, and Alan M. Wing. Active touch sensing. *Phil. Trans. R. Soc. B*, 366(1581):2989–2995, Nov 2011. ISSN 1471-2970. doi: 10.1098/rstb.2011.0167.

[5] Marten Björkman, Yasemin Bekiroglu, Virgile Högman, and Danica Kragic. Enhancing visual perception of shape through tactile glances. In *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3180–3186. IEEE, 2013.

[6] Edward Smith, David Meger, Luis Pineda, Roberto Calandra, Jitendra Malik, Adriana Romero Soriano, and Michal Drozdzal. Active 3D Shape Reconstruction from Vision and Touch. *Advances in Neural Information Processing Systems*, 34:16064–16078, 2021.

[7] A. Boehm, T. Schneider, B. Belousov, A. Kshirsagar, L. Lin, K. Doerschner, K. Drewing, C.A. Rothkopf, and J. Peters. What Matters for Active Texture Recognition With Vision-Based Tactile Sensors. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2024.

[8] Sascha Fleer, Alexandra Moringen, Roberta L Klatzky, and Helge Ritter. Learning efficient haptic shape exploration with a rigid tactile sensor array. *PloS one*, 15(1):e0226880, 2020.

[9] Jingxi Xu, Shuran Song, and Matei Ciocarlie. Tandem: Learning joint exploration and decision making with tactile sensors. *IEEE Robotics and Automation Letters*, 7(4):10391–10398, 2022.

[10] Cristiana De Farias, Naresh Marturi, Rustam Stolkin, and Yasemin Bekiroglu. Simultaneous tactile exploration and grasp refinement for unknown objects. *IEEE Robotics and Automation Letters*, 6(2):3349–3356, 2021.

[11] Volodymyr Mnih, Nicolas Heess, Alex Graves, et al. Recurrent models of visual attention. *Advances in neural information processing systems*, 27, 2014.

[12] Zhengkun Yi, Roberto Calandra, Filipe Veiga, Herke van Hoof, Tucker Hermans, Yilei Zhang, and Jan Peters. Active tactile object exploration with gaussian processes. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4925–4930. IEEE, 2016.

[13] Stanimir Dragiev, Marc Toussaint, and Michael Gienger. Uncertainty aware grasping and tactile exploration. In *2013 IEEE International conference on robotics and automation*, pages 113–119. IEEE, 2013.

[14] Amir-Hossein Shahidzadeh, Seong Jong Yoo, Pavan Mantripragada, Chahat Deep Singh, Cornelia Fermüller, and Yiannis Aloimonos. Actexplore: Active tactile exploration on unknown objects. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3411–3418. IEEE, 2024.

[15] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International conference on machine learning*, pages 1861–1870. PMLR, 2018.

[16] Aditya Bhatt, Daniel Palenicek, Boris Belousov, Max Argus, Artemij Amiranashvili, Thomas Brox, and Jan Peters. CrossQ: Batch Normalization in Deep Reinforcement Learning for Greater Sample Efficiency and Simplicity. *arXiv preprint arXiv:1902.05605*, 2019.

[17] Dosovitskiy Alexey. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv: 2010.11929*, 2020.

[18] Qiang Li, Oliver Kroemer, Zhe Su, Filipe Fernandes Veiga, Mohsen Kaboli, and Helge Joachim Ritter. A review of tactile information: Perception and action through touch. *IEEE Transactions on Robotics*, 36(6):1619–1634, 2020.

[19] Hanna Yousef, Mehdi Boukallel, and Kaspar Althoefer. Tactile sensing for dexterous in-hand manipulation in robotics—A review. *Sensors and Actuators A: Physical*, 167(2):171–187, 2011. ISSN 0924-4247. doi: https://doi.org/10.1016/j.sna.2011.02.038. Solid-State Sensors, Actuators and Microsystems Workshop.

[20] Wenzhen Yuan, Siyuan Dong, and Edward H Adelson. Gelsight: High-resolution robot tactile sensors for estimating geometry and force. *Sensors*, 17(12):2762, 2017.

[21] Mike Lambeta, Po-Wei Chou, Stephen Tian, Brian Yang, Benjamin Maloon, Victoria Rose Most, Dave Stroud, Raymond Santos, Ahmad Byagowi, Gregg Kammerer, et al. Digit: A novel design for a low-cost compact high-resolution tactile sensor with application to in-hand manipulation. *IEEE Robotics and Automation Letters*, 5(3):3838–3845, 2020.

[22] John Lloyd and Nathan F Lepora. Pose-and-shear-based tactile servoing. *The International Journal of Robotics Research*, 43(7):1024–1055, 2024.

[23] Yi Zheng, Filipe Fernandes Veiga, Jan Peters, and Veronica J Santos. Autonomous learning of page flipping movements via tactile feedback. *IEEE Transactions on Robotics*, 38(5):2734–2749, 2022.

[24] Zhao-Heng Yin, Binghao Huang, Yuzhe Qin, Qifeng Chen, and Xiaolong Wang. Rotating without seeing: Towards in-hand dexterity through touch. *arXiv preprint arXiv:2303.10880*, 2023.

[25] Haozhi Qi, Brent Yi, Sudharshan Suresh, Mike Lambeta, Yi Ma, Roberto Calandra, and Jitendra Malik. General in-hand object rotation with vision and touch. In *Conference on Robot Learning*, pages 2549–2564. PMLR, 2023.

[26] Dominik Bauer, Zhenjia Xu, and Shuran Song. Doughnet: A visual predictive model for topological manipulation of deformable objects. In *European Conference on Computer Vision*, pages 92–108. Springer, 2025.

[27] Yizhou Chen, Andrea Sipos, Mark Van der Merwe, and Nima Fazeli. Visuo-tactile transformers for manipulation. *arXiv preprint arXiv:2210.00121*, 2022.

[28] Baojiang Li, Jibo Bai, Shengjie Qiu, Haiyan Wang, and Yuting Guo. VITO-transformer: a visual-tactile fusion network for object recognition. *IEEE Transactions on Instrumentation and Measurement*, 2023.

[29] Sergey Ioffe. Batch renormalization: Towards reducing minibatch dependence in batch-normalized models. *Advances in neural information processing systems*, 30, 2017.

[30] Anurag Arnab, Mostafa Dehghani, Georg Heigold, Chen Sun, Mario Lučić, and Cordelia Schmid. Vivit: A video vision transformer. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 6836–6846, 2021.

[31] Tim Schneider, Cristiana de Farias, Roberto Calandra, Liming Chen, and Jan Peters. The Tactile MNIST Benchmark. `https://sites.google.com/robot-learning.de/tactile-mnist`, 2025.

[32] Michal Nauman, Mateusz Ostaszewski, Krzysztof Jankowski, Piotr Miłoś, and Marek Cygan. Bigger, Regularized, Optimistic: scaling for compute and sample-efficient continuous control. In *Advances in neural information processing systems*, 2024.

[33] Hojoon Lee, Dongyoon Hwang, Donghu Kim, Hyunseung Kim, Jun Jet Tai, Kaushik Subramanian, Peter R Wurman, Jaegul Choo, Peter Stone, and Takuma Seno. SimBa: Simplicity Bias for Scaling Up Parameters in Deep Reinforcement Learning. *arXiv preprint arXiv:2410.09754*, 2024.

[34] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A. Efros. Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, Oct 2017.

[35] Alexander I Cowen-Rivers, Wenlong Lyu, Rasul Tutunov, Zhi Wang, Antoine Grosnit, Ryan Rhys Griffiths, Alexandre Max Maraval, Hao Jianye, Jun Wang, Jan Peters, et al. Hebo: Pushing the limits of sample-efficient hyper-parameter optimisation. *Journal of Artificial Intelligence Research*, 74:1269–1349, 2022.

[36] Zilin Si and Wenzhen Yuan. Taxim: An example-based simulation model for gelsight tactile sensors. *IEEE Robotics and Automation Letters*, 7(2):2361–2368, 2022.

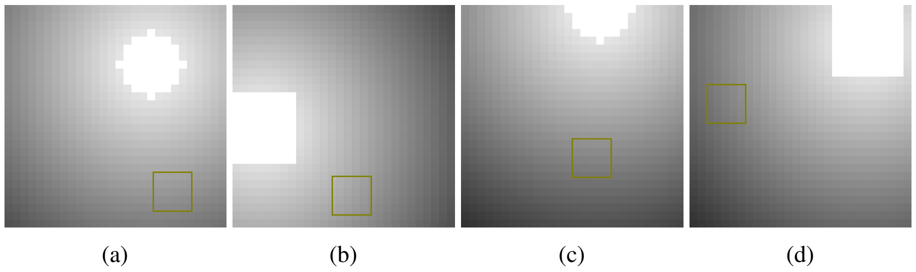

Figure 7: Episode starting conditions of the Circle-Square task. The agent's glimpse and the object (circle (a, c) or square (b, d)) are placed in random locations on the field. Besides the color gradient, the agent receives no information about the object's location on the field.

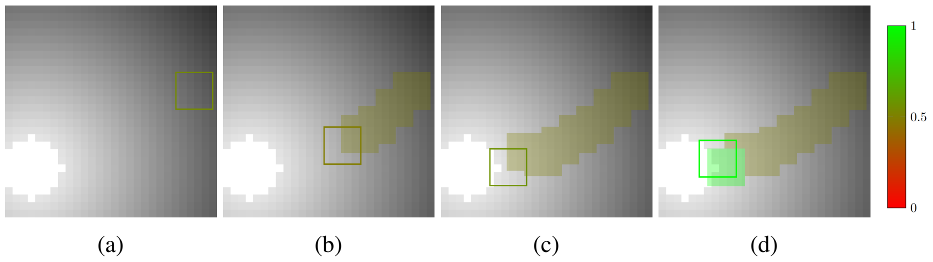

Figure 8: Visualization of an episode in the *Circle-Square* task. (a) The agent starts at a random location and uses the color gradient to locate the object. It can only observe a 5×5 pixel patch. (b) The agent follows the gradient, gradually gathering information. Without full certainty, it predicts a 50/50 probability between classes along the way. Colored boxes show past glances, with color indicating prediction confidence. (c) The agent reaches the object at the corner. (d) Upon confident identification, the agent classifies the object as a square (bright green box) and maintains this prediction in later steps.

Appendix

A Environment Details

Here, we detail each of the tasks that were evaluated. That is the Circle-Square 2D classification task, the Tactile MNIST tactile classification task, the Starstruck counting task, and the Toolbox pose estimation task.

A.1 Circle-Square Task

In each Circle-Square episode, the agent receives a 28×28 grayscale image containing either a circle or a square. It can only observe a 5×5 pixel region (glimpse), with the initial location randomized. A color gradient in the background helps guide exploration, but the agent has no access to the object's position. See Fig. 7 and Fig. 8 for illustration of the Circle-Square task.

Actions $a_t \in [-1,1]^2$ are mapped to pixel motion as $a_t \cdot 5.6\,\text{px}$ (20% of the image width), allowing smooth movement across the image. We use bilinear interpolation to compute the glimpse values even at non-integer positions. The agent's prediction $y_t \in \mathbb{R}^2$ is interpreted as logits for circle vs. square:

$$p_{\text{circle}}(y_t) = \frac{e^{y_t^{(1)}}}{e^{y_t^{(1)}} + e^{y_t^{(2)}}}, \quad p_{\text{square}}(y_t) = \frac{e^{y_t^{(2)}}}{e^{y_t^{(1)}} + e^{y_t^{(2)}}}.$$

Cross-entropy loss is used for training:

$$\ell(y_t^*, y_t) = -\sum_{c \in \{\text{circle,square}\}} \delta(y_t^*, c) \log\left(p_c(y_t)\right).$$

We apply a regularizing reward penalty on the magnitude of each action: $r(h_t, a_t) = 10^{-3}\|a_t\|^2$.

Due to the small input size, we do not use a vision encoder; instead, the input is directly flattened into a 25-dimensional vector. This design choice allows a fairer comparison to HAM [8], which also operates on flat image data,

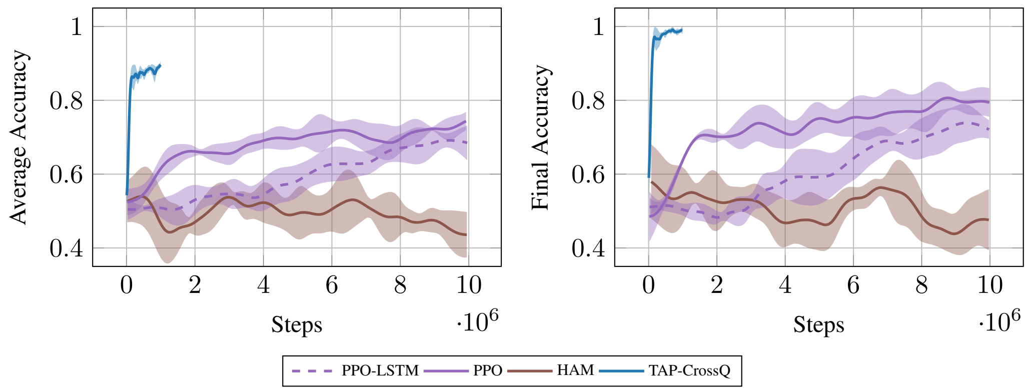

Figure 9: Additional experiments on the Circle-Square environment, comparing HAM with two PPO-based variants: PPO−TAP and PPO−LSTM. The difference between PPO−TAP and PPO−LSTM is that PPO−TAP uses our transformer model, while PPO−LSTM uses HAM's LSTM model. TAP−CrossQ's run with 250K steps is shown for reference. The left plot shows the average prediction accuracies, while the right plot shows the final prediction accuracies. Training is terminated after 10M environment steps.

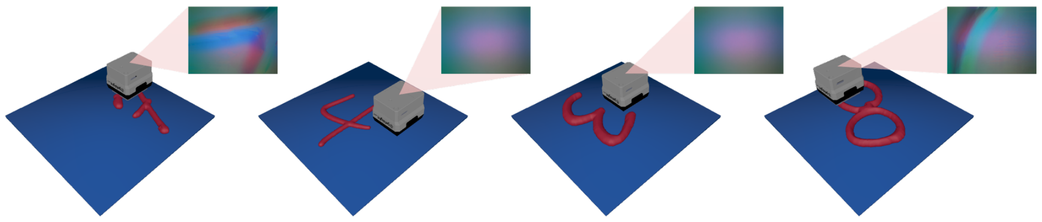

Figure 10: The simulated *Tactile MNIST* classification benchmark [31], which we use for evaluating our method. The objective of the Tactile MNIST task is to identify the numeric value of the presented digit by touch only. In every step, the agent decides how to move the finger and predicts the class label. The haptic glance is computed via the Taxim [36] tactile simulator.

Figure 9 shows an additional experiment on Circle-Square, where we compare HAM to two PPO-based variants, one using HAM's LSTM model (PPO−LSTM) and one using our transformer model (PPO−TAP). Note that, unlike in the experiment shown in Fig. 4 where we stopped training after 250K environment steps, here, we let the training run for 10M steps. Despite the longer training time, HAM fails to achieve a final prediction accuracy that is better than random guessing. The PPO variant using HAM's model (PPO−LSTM) achieves a final prediction accuracy of 72% after 10M steps, while the PPO variant with our model (PPO−TAP) achieves around 79%. In addition to the difference in final performance of PPO−LSTM and PPO−TAP being 7%, PPO−TAP improves much quicker in the beginning than PPO−LSTM. All hyperparameters for this experiment were again tuned using HEBO [35] and each method was trained from scratch on 5 seeds.

These results, in conjunction with the results shown in Section 4.1, indicate that HAM's issues in solving the Circle-Square task may have two reasons: First, HAM's LSTM model seems to be ill-suited for learning this task, as all other compared methods (TAP−SAC, TAP−CrossQ, and PPO−TAP) each performed worse with HAM's LSTM model than with our transformer model. Second, the fact that PPO and HAM are both on-policy algorithms likely has a negative impact on their sample efficiency, as samples collected from the environment cannot be reused later on. Off-policy algorithms, such as TAP−SAC and TAP−CrossQ, on the other hand, store samples over the course of the entire training and revisit them many times, making optimal use of the information gathered during training. We see this effect clearly in Fig. 9, where TAP−CrossQ outperforms PPO−TAP by orders of magnitude in sample-efficiency, despite both algorithms using the same model for their policy and class predictions.

A.2 Tactile MNIST Benchmark

In the Tactile MNIST benchmark (see 10), the agent is presented with a 3D model of a high-resolution MNIST digit, placed randomly on a 12×12cm plate. Each digit is up to 10cm in width and height.

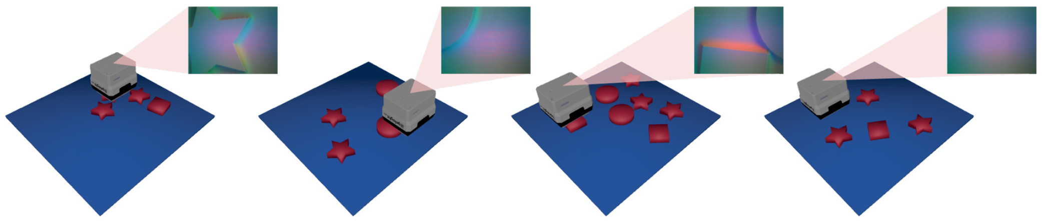
Figure 11: The simulated *Starstruck* [31] task, which we use for evaluating our method. The objective of the Starstruck task is to count the number of stars among the presented objects on the plate. In every step, the agent decides how to move the finger and predicts the number of stars. The haptic glance is computed via the Taxim [36] tactile simulator.

The agent uses a single simulated GelSight Mini sensor [20] to explore the plate surface. The sensor outputs a 32×32 pixel image rendered from a depth map by Taxim [36]. Additionally, the agent receives the 3D position of the sensor as input.

Each episode begins with a randomly selected digit, randomly placed and oriented. The agent has 16 steps to explore and classify the object. At each step, it chooses a movement action $a_t \in [-1, 1]^2$, which corresponds to a translation of up to 1.5cm per axis. The sensor is automatically positioned to maintain a 2mm indentation into the 4mm-thick gel. To simulate the object shifting around when being manipulated by the agent, we apply Gaussian random noise to the position and orientation of the object throughout the episode.

The classification output is a 10-dimensional logit vector. A standard cross-entropy loss is used:

$$\ell(y_t^*, y_t) = -\sum_{c=1}^{10} \delta\left(y_t^*, c\right) \log\left(p_c\left(y_t\right)\right), \quad p_c\left(y_t\right) = \frac{e^{y_t^{(c)}}}{\sum_{i=0}^{10} e^{y_t^{(i)}}}.$$

The reward is used only for regularizing motion: $r(h_t, a_t) = 10^{-3}\|a_t\|^2$.

We train `TAP-SAC`, `TAP-CrossQ`, and `TAP-RND` from scratch (no pre-trained encoders), using 5 random seeds for 5M steps. Hyperparameters are tuned via the `HEBO` Bayesian optimizer. `HAM` [8] is not evaluated here as it lacks an image encoder. Evaluation is done using digits not seen during training.

A.3 Starstruck Task

In the Starstruck task (see 11), each episode begins with a random configuration of geometric objects—circles, squares, and between one to three identical 3D star shapes—scattered across a 12×12cm plate. The agent explores the scene using a simulated GelSight Mini sensor, identical to that used in Tactile MNIST, providing a 32×32 tactile image, while also receiving the 3D sensor position as input.

Actions $a_t \in [-1, 1]^2$ again correspond to a maximum motion of 1.5cm per axis. The sensor maintains a fixed 2mm indentation into the gel on each step. Unlike Tactile MNIST, the label in each episode is the total number of stars present (1, 2, or 3), and a prediction is only considered correct if it exactly matches the ground truth.

Each agent – `TAP-SAC`, `TAP-CrossQ`, and `TAP-RND` – is trained from scratch using 5 random seeds for 5M steps. We reuse the hyperparameters optimized on Tactile MNIST without modification to evaluate robustness across tasks. Evaluation is performed on held-out shape configurations, which are not seen during training.

A.4 Toolbox Task

In the Toolbox task (see 12), the agent has to estimate the pose of a 24cm long wrench that is placed in a uniformly random 2D position and orientation on a 30×30cm plate. Similarly to the previous two tasks, the agent explores the scene using a simulated GelSight Mini sensor, providing a 32×32 tactile image, while also receiving the 3D sensor position as input. Actions $a_t \in [-1, 1]^2$ again correspond to a maximum motion of 1.5cm per axis, and the sensor maintains a fixed 2mm indentation into the

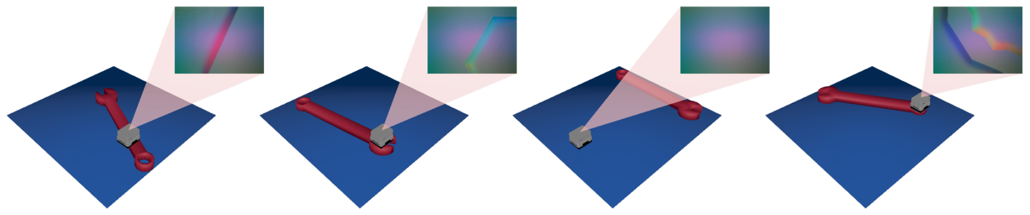

Figure 12: The simulated *Toolbox* task, which we use for evaluating our method. The objective of the Toolbox task is to determine the 2D pose (2D position and orientation angle) of the object relative to the platform center. In every step, the agent decides how to move the finger and predicts the 2D pose. The haptic glance is computed via the Taxim [36] tactile simulator.

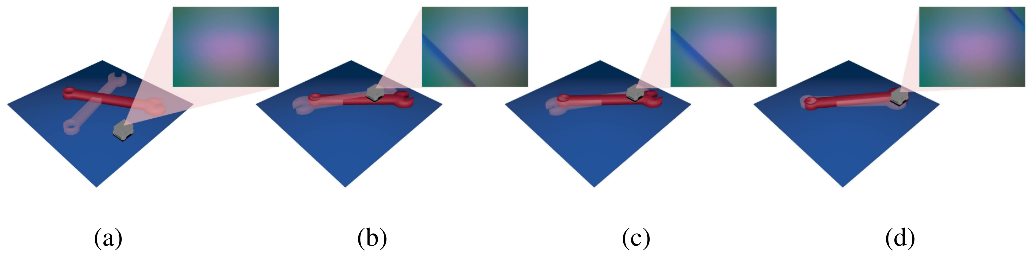

Figure 13: Exploration strategy learned by our `TAP-CrossQ` agent. In the beginning (a), both sensor and wrench start in uniformly random places on the platform. The agent guesses a central position of the wrench (illustrated by the transparent wrench) to minimize error in the absence of any further information. To find the object efficiently, the agent has learned a circular search pattern and therefore quickly locates the object (b). However, the information it currently has is not enough, as the orientation of the wrench is not clear by just touching the handle. Thus, it randomly guesses the wrong orientation, with the open jaw pointing left instead of right. To gather information, it moves along the handle (c) until it finds the open jaw (d) and immediately corrects the angle of it pose estimation.

gel on each touch. To simulate the object shifting around when being manipulated by the agent, we apply Gaussian random noise to the position and orientation of the object throughout the episode.

The Toolbox task poses two challenges: finding the object and determining its exact position and orientation. Once the object is found, determining its orientation is still not trivial, as many touch locations only provide ambiguous data. Hence, as shown in Fig. 13, even after the object is found, an exploration strategy for determining its pose must be executed.

Each agent – `TAP-SAC`, `TAP-CrossQ`, and `TAP-RND` – is trained from scratch using 5 random seeds for 10M steps. Similar to the Starstruck task, we reuse the hyperparameters optimized on Tactile MNIST without modification to evaluate robustness across tasks.