2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
October 14-18, 2024. Abu Dhabi, UAE

# Learning When to Stop: Efficient Active Tactile Perception with Deep Reinforcement Learning

*Christopher Niemann[1,†], *David Leins[2], Luca Lach[3] and Robert Haschke[4]

***Abstract*— Actively guiding attention is an important mechanism to employ limited processing resources efficiently. The Recurrent Visual Attention Model (RAM) has been successfully applied to process large input images by sequentially attending to smaller image regions with an RL framework. In tactile perception, sequential attention methods are required naturally due to the limited size of the tactile receptive field. The concept of RAM was transferred to the haptic domain by the Haptic Attention Model (HAM) to iteratively generate a fixed number of informative haptic glances for tactile object classification. We extend HAM to a system capable of actively determining when sufficient haptic data is available for reliable classification. To this end, we introduce a hybrid action space, augmenting the continuous glance location with the discrete decision of when to classify. This allows balancing the cost of obtaining new samples against the cost of misclassification, resulting in an optimized number of glances while maintaining reasonable accuracy. We evaluate the efficiency of our approach on a handcrafted dataset, which allows us to compute the most efficient glance locations.**

## I. INTRODUCTION

Active object exploration based on tactile feedback has become a popular area of research in robotics [1]. Such strategies generally consist of an exploratory behavior that produces a sequence of tactile data, here also referred to as haptic glances, and a classification model. Most approaches perform simple object classification, while some also attempt to estimate object properties such as stiffness, material, shape, or pose. Early works base their exploration policy on Bayesian inference [2], [3], [4] or Gaussian processes [5]. The decision of whether the sequence contains sufficient information for a satisfactory classification or if additional glances are required is made based on model certainty or the potential information gained from another glance. This introduces the manual identification and potential tuning of fixed threshold parameters. More recent approaches apply deep learning models to the task of tactile exploration. In [6], three different networks process the tactile features, producing a probability distribution over object classes. Instead of defining a stopping criterion, the authors execute a fixed number of one to five glances and combine their embeddings for the final classification. Their results show that near-perfect classification accuracy was achieved with only two glances, rendering further glances superfluous. In [7], Fleer et al. introduce the Haptic Attention Model (HAM), a reinforcement learning framework for the exploration of haptic shapes for classification. Using a sequence model, HAM iteratively integrates the acquired haptic information, predicts the next best glance location and finally performs the classification after a fixed number of steps.

In this work, we directly integrate the decision of when to stop the exploration into our model. We thereby tackle the aforementioned problems: Instead of performing a fixed number of glances, the model can dynamically stop the exploration based on the currently available haptic information. The model itself decides when it has gathered sufficient data to reliably classify the object at hand to avoid unnecessary glances. This is highly important in scenarios where performing glances is costly. Furthermore, we alleviate the need for manual tuning of threshold parameters to stop the exploration.

We adopt the structure of HAM and split the RL policy into two distinct sub-policies: one proposes the next glance location while the other one decides whether to classify the object using the current glance sequence or to keep exploring. The policy penalizes incorrect classifications and only classifies once it has gathered enough tactile data for correct classification. Additional glances also incur a minor penalty for the policy as an incentive to avoid unnecessary glances and to keep the sequence length minimal. We evaluate our method in simulation on a synthetic set of objects, for which we can compute the optimal sequence of glances. This allows us to assess whether our proposed method can converge to the theoretical minimum number of required glances. In summary, our main contribution is reducing the number of glances to an optimum while preserving classification accuracy by integrating the decision of whether to continue exploration or perform classification directly into the ML model. This approach is not limited to the haptic domain, but can be applied to any explorative reinforcement learning task.

The supplementary source code is available at `https://github.com/cniemann0/haptic-exploration`.

## II. RELATED WORK

In this section we describe the concept of the Haptic Attention Model (HAM) and the model it is derived from, the Recurrent Attention Model (RAM).

### A. Recurrent Attention Model

The fundamental idea of RAM [8] is to reduce the computational cost of processing large images or video sequences

*Equal contribution
†Supported by the BMDV DZM enableATO project (19DZ23002B)
[1]CoR-Lab, CITEC, Bielefeld University, Germany
[2]Machine Learning Group, CITEC, Bielefeld University, Germany
[3]Institut de Robòtica i Informàtica Industrial, CSIC-UPC
[4]Social Cognitive Systems Group, CITEC, Bielefeld University, Germany

with CNNs by identifying salient regions. These regions are processed with high resolution, while more distant, less salient regions, are encoded and processed with a progressively lower resolution. An LSTM sequentially processes *glimpses* of the input at different locations and integrates the acquired information to achieve a task-specific goal, e.g., image-based classification or tracking. This process is guided by a network that generates the next glimpse location based on the current state encoded by the LSTM. Mnih et al. [8] choose to model the optimization of generated glimpses as an RL problem, where the reward is based on the task's success, which depends on the set of glimpses. In the case of a classification task, the classifier is optimized with the supervised cross-entropy loss, while the glimpse action is optimized with the REINFORCE algorithm.

### B. Haptic Attention Model

Fleer et al. propose HAM [7], an RL framework inspired by RAM, to learn the exploration of four primitive shapes with a square tactile sensor array, the Myrmex sensor [9]. Transferring the RAM concept to the haptic domain, they learn the generation of *haptic glances* for object classification. Haptic glances are brief, tactile object interactions that are spatially and temporally constrained [10]. The sensor with a size of $8 \times 8$ cm yields a $16 \times 16$ pressure matrix. It is mounted to a robot arm that sequentially performs haptic glances in a predefined exploration area containing the object. HAM sequentially integrates the obtained haptic information and guides the exploration by generating the next glance location. After a fixed number of glances $N$, the exploration ends, and the object is classified.

For the exploration task, the object is placed in the center of an area that defines the admissible glance locations. Before performing a glance, the sensor is positioned above the exploration area at an initial pose, determined by the glance parameters. The downward-facing sensor is then lowered until contact with the object occurs or the $z$-height reaches a lower limit, yielding a pressure pattern $M \in \mathbb{R}^{16\times16}$. Due to the nature of the primitive objects used, [7] restrict the initial pose to two degrees of freedom: the $x$-position and the angle of rotation $\varphi$ around the $y$-axis.

HAM adopts the structure of RAM, which consists of multiple sub-networks performing different tasks. For each newly obtained glance at time step $t$, an embedding network encodes the pressure $M_t$ and the glance parameters $x_t, \varphi_t$ into a unified glance embedding. An LSTM iteratively processes these features and computes a latent sequence representation $h_t$ encoding the haptic information of the sequence of glances. At each step, $h_t$ is used to generate the next glance parameters $a_{t+1} = (x_{t+1}, \varphi_{t+1})$. This process is repeated for $N$ steps before predicting the object class based on the final sequence representation $h_N$. The initial glance parameters $a_1$ for the first glance are generated randomly.

While the approach of [7] learns to generate *any* sequence of parameters that allow accurate classification, it does not directly enforce the most efficient exploration. Due to the static hyperparameter $N$, the exploration is always of fixed length. In an unbalanced object set, some objects might be distinguishable immediately within only a few glances due to their unique shape. Meanwhile, a specific subset of objects might be very similar to each other (e.g. cylindrical objects such as glasses, bottles and cans) and, thus, require more glances. $N$ can only be chosen per object set, which inevitably results in either unnecessary glances for unique objects or unsatisfactory classification results for hardly distinguishable objects. Thus, we propose to increase the exploration efficiency by letting the RL agent dynamically decide when it has gathered enough tactile information for accurate classification.

## III. METHODS

In this section, we detail the changes we made to the RL process with regard to the previous work we built on. We introduce the concept of hybrid actions, their integration into the RL policy, and necessary changes to the reward function to incentivize optimizing the number of haptic glances.

### A. RL Modifications for Dynamic Exploration

Our approach builds upon the RL methodology of [7]. In order to perform a varying number of explorative glances, we propose two fundamental modifications:

1) **Decision Action**: In addition to the glance location, the model generates a discrete decision whether to perform another glance or stop and classify the object. Whenever the model chooses classification, the RL episode ends.
2) **Glance Penalty**: We introduce a small negative reward for each glance to incentivize the model to take as few glances as possible.

We consider the general case where the continuous glance pose is parameterized by actions $a_g \in A_{\text{glance}} := \mathbb{R}^d$. We introduce an additional discrete action $a_d \in A_{\text{decision}} := \{\text{GLANCE}, \text{STOP}\}$ and formally define the resulting *hybrid discrete-continuous* action space as the combination of both sub-spaces:

$$A_{\text{hybrid}} = A_{\text{decision}} \times A_{\text{glance}}$$

The tuple $a = (a_{\text{d}}, a_{\text{g}})$ represents an action containing both the stopping decision $a_{\text{d}}$ and the glance parameters $a_{\text{g}}$. This action definition allows for dynamic exploration that ends as soon as $a_{\text{d}} = \text{STOP}$. In case $a_{\text{d}} = \text{GLANCE}$, the next glance is determined by the glance parameters $a_{\text{g}}$. We define an upper limit $N_{\text{max}}$ to prevent infinite exploration. The episode ends whenever classification is performed or when the number of glances exceeds $N_{\text{max}}$. With this action space, the RL agent can actively terminate the exploration at any time.

Delegating the stopping decision to the model requires an incentive to minimize the number of glances. We integrate this objective into the reward function by associating a cost with performing a glance. To this end, we introduce a penalty $p < 0$ for every glance during exploration as a negative immediate reward. This idea was already mentioned by Mnih

et al. [8], however, it was not implemented in RAM or HAM. This results in the following reward at time step $t$:

$$r_t = \begin{cases} p & \text{if } a_\mathrm{d} = \text{GLANCE and } t \leq N_{max} \\ r_c & \text{otherwise} \end{cases}$$

Given the predicted object class $\hat{c}$, and the ground truth class $c$, we define the classification reward $r_c$ as

$$r_c = \begin{cases} r_{c+} & \text{if } \hat{c} = c \\ r_{c-} & \text{otherwise} \end{cases}$$

where $r_{c+} = 1$ and $r_{c-} = -1$. As the main objective is accurate classification, we need to ensure that the penalty introduced does not outweigh the final classification reward. A correct classification after $N_{max}$ glances should result in a higher reward than an instant misclassification. Formally, we define this requirement as

$$\begin{aligned} & r_{c+} + N_{max} \cdot p > r_{c-} \qquad \big| \, r_{c+} = -r_{c-} \\ \Leftrightarrow \quad & 0 > p > -\frac{2r_{c+}}{N_{max}} \end{aligned} \tag{1}$$

For convenience, we introduce a *balancing factor* $b \in (0, 1)$ that determines p as the fraction of this lower limit:

$$p = -\frac{2r_{c+}}{N_{max}} \cdot b \tag{2}$$

### B. Training Algorithm

The fundamental training methodology directly transfers from [7]. We employ the same method of training the policy with RL using the reward signal $r_c$ from the classification output. This requires the simultaneous optimization of the policy and the classification output.

*1) Policy Optimization:* To learn a policy that generates efficient glance parameters $a_g$ and decisions $a_d$, we use the RL algorithm *REINFORCE with baseline* [11], which is also employed by RAM and HAM. It represents the basic approach behind the family of RL algorithms called policy gradient methods, which can handle discrete and continuous action spaces. In contrast to value-based methods that derive a policy from learned value estimates, the policy is modeled as a *parameterized function* $\pi_\theta$ [12]. We denote the policy parameters as $\theta$. By repeatedly querying the policy $\pi_\theta$, a trajectory of states $s_t$, actions $a_t$ and rewards $r_t$ is generated:

$$\tau = (s_1, a_1, r_1, \ldots, s_T, a_T, r_T)$$

The goal is to maximize a scalar objective function that measures the expected cumulative reward

$$J(\theta) = \mathbb{E}_{\pi_\theta}\left[\sum_{t=1}^{T} \gamma^{t-1} r_t\right]$$

where $\gamma \in (0, 1]$ is a discount factor. The policy parameters $\theta$ (i.e., the network weights) are directly optimized via gradient ascent for policy gradient methods [13]:

$$\theta_{t+1} = \theta_t + \alpha_\theta \nabla_\theta J(\theta_t) \tag{3}$$

REINFORCE with baseline approximates the gradient $\nabla J(\theta)$ with an expectation under $\pi$, which we can then sample from by following $\pi$ using Monte-Carlo methods [13]:

$$\nabla J(\theta) \propto \mathbb{E}_\pi\left[(G_t - v_w(s_t)) \nabla \ln \pi_\theta(a_t|s_t)\right]$$

Here, $G_t = \sum_{k=t}^{T} \gamma^{k-t} r_t$ is the discounted return obtained from a sampled trajectory after taking action $a_t$ in state $s_t$. The baseline $v_w(s_t)$ is an estimate of the expected discounted return of the current policy $\pi_\theta$ when starting in state $s_t$, also known as the state-value $V^\pi(s_t)$. Similarly to the policy, $v_w$ is a parameterized function with parameters $w$. It is optimized with the MSE loss to minimize the error between the actual returns $G_t$ and the estimated returns $v_w(s_t)$. By including a state-dependent baseline, we reduce the variance of the gradient estimate while remaining unbiased [14].

Similar to [7], we train the classifier to predict the correct object based on the glance sequences obtained from on-policy RL using the cross-entropy loss.

*2) Hybrid Policy:* We employ two separate sub-policies, $\pi_\mathrm{d}$ and $\pi_g$, to generate both the discrete decision action and the continuous glance action. The decision policy $\pi_g$ generates probabilities by generating scores for each discrete decision $a_d \in A_{\text{decision}}$ and applying the softmax function. We can then draw the action $a_d \sim \pi_d(\cdot|s)$ from the resulting probability distribution. The glance policy $\pi_g$ represents a multivariate normal distribution with means $\mu(s) \in \mathbb{R}^d$ and standard deviations $\sigma \in \mathbb{R}^d$ over continuous glance parameters $a_g \in \mathbb{R}^d$. While the means are state-dependent outputs of the policy network, we learn a state-independent vector of standard deviations. To ensure that $\sigma$ is always greater than zero, we learn $\ln \sigma$ and apply the exponential function to obtain $\sigma$. The continuous glance policy $\pi_g$ is then defined as:

$$\pi_g(a_g|s) = \prod_{i=1}^{d} \mathcal{N}(a_{g,i}|\mu_i(s), \sigma_i)$$

The resulting hybrid policy is the joint distribution of both sub-actions [15]:

$$\pi(a|s) = \pi(a_\mathrm{d}, a_\mathrm{g}|s) = \pi_\mathrm{d}(a_\mathrm{d}|s)\, \pi_\mathrm{g}(a_\mathrm{g}|s) \tag{4}$$

Policy gradient methods inherently compute $\nabla \ln \pi(a|s)$ to calculate the gradient of the objective function $\nabla J(\theta)$, hence our method can be integrated into any policy gradient method (such as PPO [16] or TRPO [17]) by inserting Eq. 4, yielding

$$\nabla \ln \pi_\mathrm{d}(a_\mathrm{d}|s)\, \pi_\mathrm{g}(a_\mathrm{g}|s) = \nabla \ln \pi_\mathrm{d}(a_\mathrm{d}|s) + \nabla \ln \pi_\mathrm{g}(a_\mathrm{g}|s)$$

Fig. 1 illustrates the hybrid action generation process.

### C. Model Architecture

We propose an abstract model architecture based on HAM, representing a general framework for haptic exploration. It can be adapted by employing task-specific networks.

As input to their sequence model, [7] embed each tactile pattern $M_t$ together with the glance parameters $x_t, \varphi_t$. Instead, we embed $M_t$ with the final sensor pose, represented by a position $p$ and a rotation quaternion $q$. While the glance parameters only describe the initial sensor position, the final

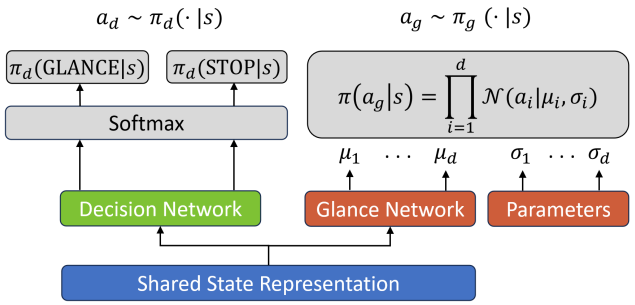



Fig. 1: Generation of the hybrid actions $a = (a_d, a_g)$ with two separate action networks based on a shared state representation input.

sensor pose can provide additional information, such as the $z$-height of the object.

The *embedding network* $f_e : (M, p, q) \mapsto g$ maps a glance – defined by the obtained tactile pattern $M$, position $p$ and orientation $q$ – to an embedded representation $g$. The *sequence model* $f_s : g_1, \ldots, g_t \mapsto h_t$ extracts relevant features from the sequence of embedded haptic glances and encodes them into a fixed-size latent sequence representation $h_t$. In both HAM and RAM, an LSTM was used to process the sequential input. In this work, we additionally consider Transformers.

The latent sequence representation $h_t$ produced by the *sequence model* acts as input for both the classification and action generation: The *classification network* $f_c : h_t \mapsto \hat{c}$ predicts the object class $\hat{c}$, and *the decision network* $f_d : h_t \mapsto a_d$ and *glance network* $f_g : h_t \mapsto a_g$ represent the policy by generating the action $a_t = (a_d, a_g)$. The *value network* $f_v : h_t \mapsto v$ estimates the state-value $v(s)$ required for the RL training phase. The entire model architecture is depicted in Fig. 2.

The hyperparameter $d_{\text{hidden}}$ specifies the dimension of the embedded haptic glances $g$ and the latent sequence representation $h$. Thus, it should be chosen according to the complexity of the task and the sensor morphology.

While [7] start the exploration with random glance parameters, we also allow the model to learn the best pose for the first glance. As the model can not work on an empty sequence input, i.e., it requires a previous glance to generate an action, we introduce a learnable embedding $g_0$ that represents the sequence's start, which allows us to start the input sequence with $g_0$ to generate an initial action.

### D. Classifier Pre-Training

Training our proposed architecture end-to-end from scratch is be inefficient because the simultaneous optimization of classification and glance generation is interdependent. The classifier can only learn to identify objects accurately from informative glances obtained by the glance policy, and the glance model can only learn informative glances from the reward signal of an accurate classifier. To mitigate this problem, we pre-train the classifier on glance sequences of varying length with random glance poses uniformly sampled within the exploration area. Since the classification network and the policy networks both depend on the output of the

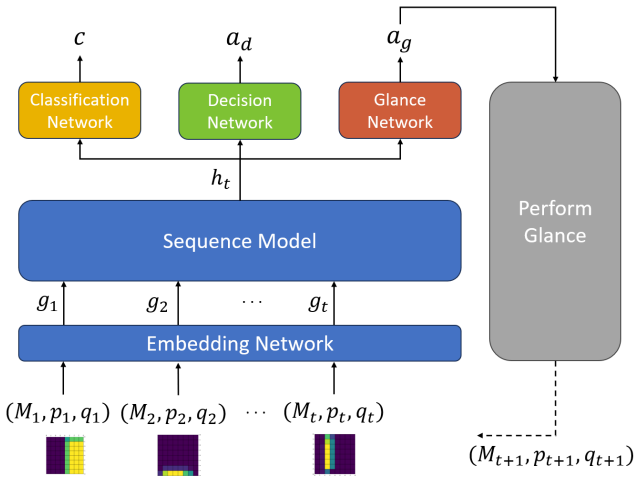



Fig. 2: The model architecture and data flow. The haptic glances are embedded and fed into a sequence model that generates a latent sequence representation containing the haptic information required for both classification and action generation. We omit the state-value network for brevity.

sequence model, this pre-training not only speeds up the RL training because a useful reward signal is available from the very beginning, but also because the embedding and sequence models are only fine-tuned and not learned from scratch.

## IV. EXPERIMENTS

We perform exploration with the experimental setup from [7] and a fixed number of glances using our adapted model. We find that even a single glance generated by our model using the initial embedding $g_0$ achieves a classification accuracy of **98**% for both the LSTM and Transformer. [7] report an accuracy of **54**% for the first glance (with randomly generated parameters) which increases to **83**%, **91**%, and **97**% for each consecutively added glance. The very high accuracy of our model for only a single glance implies that the object primitives used in [7] are too simple to discriminate, rendering the object set unsuitable for examining the efficiency of the exploration.

Thus, we perform a proof-of-concept study with a complex object set to assess whether our model can converge toward an optimal number of average glances while preserving classification accuracy. This requires us to be able to compute the optimal sequence of glance parameters analytically. However, this can only be achieved if glanceable features relevant for unique identification within and between items in the object set and their exact locations are known. For objects with realistic textures, computing this information is non-trivial.

Instead, we opt for a handcrafted set of synthetic objects specifically designed to allow for a simplified abstract representation of objects. Using this representation, we can control the imbalance of the object set with respect to the number of required glances by defining ambiguities in the features. Furthermore, we can compute an optimal policy that minimizes the expected number of glances and lets us benchmark the efficiency of our proposed learning framework against the

theoretical optimum. This method resembles the concept of ambiguities in an object set on the most fundamental level.

### A. Composite Object Environment

The fundamental concept of this object set is based on primitive features $F = \{0, \dots, 5\}$, shown in Fig. 3. These features, which are designed to be easily distinguishable, are *composed* to constitute a set of composite objects following a specific rule: each composite object consists of a subset of four features placed into predefined sockets. Thus, we ensure that all objects' structures are identical, but the specific features can differ. The sockets are each assigned a unique index from 0 to 3, as indicated by the left image in Fig. 4. This allows us to define an abstract representation of a composite object as a 4-tuple $(f_0, \dots, f_3)$, where the features $f_i \in F$ occur at the index of their socket. Fig. 4 illustrates the exemplary composite object $(2, 4, 1, 3)$. This compository approach allows us to flexibly define tactile similarities and differences between objects by selecting the same or dissimilar features.

In this experimental setup, we use an $8 \times 8cm$ tactile sensor array [9] to collect $16 \times 16$ tactile images. To avoid the additional complexity of controlling a kinematic chain with the sensor as an end-effector, we directly control the sensor's pose with the simulations' soft constraint system, mimicking the behavior of a cartesian impedance controller.

The feature footprints do not exceed $10 \times 10cm$, and the centers of neighboring sockets are 20cm apart. This ensures that the sensor can only acquire information about one feature per glance.

To conform to the exploration setup of the composite objects, we model the glance parameters as the $x$ and $y$-position: $a = (x, y)$. Both $x$ and $y$ are restricted by the exploration area, as indicated in Fig. 4. However, in our setup, the sensor does not support rotation as is not necessary to identify the easily distinguishable sub-features.

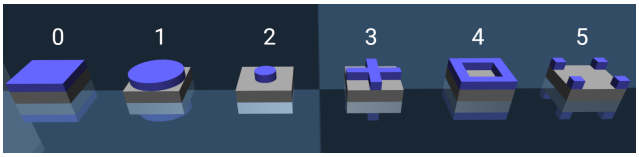



Fig. 3: Tactile features constituting composite objects.

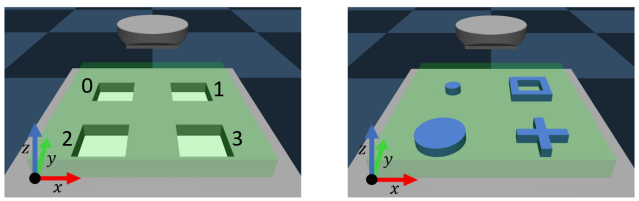



Fig. 4: Exploration setup for composite objects. The tactile sensor (grey cylinder) is floating above the exploration area (green box). On the left, the empty structure of the composite objects is shown. Each location is labelled by an index. On the right, the composite object with the abstract representation (2, 4, 1, 3) is shown.

| Composite Objects | | | | | |
|---|---|---|---|---|---|
| Index | Features | Index | Features | Index | Features |
| 0 | $(2, 2, 3, 0)$ | 5 | $(2, 2, 3, 5)$ | 10 | $(2, 3, 3, 5)$ |
| 1 | $(1, 2, 3, 5)$ | 6 | $(2, 3, 4, 1)$ | 11 | $(2, 2, 4, 5)$ |
| 2 | $(1, 2, 4, 4)$ | 7 | $(2, 3, 4, 4)$ | 12 | $(3, 3, 3, 2)$ |
| 3 | $(1, 2, 4, 5)$ | 8 | $(3, 3, 3, 4)$ | | |
| 4 | $(2, 2, 4, 4)$ | 9 | $(3, 2, 3, 2)$ | | |

TABLE I: The hand-crafted composite objects, defined by their abstract representations.

### B. Hand Crafted Object Set

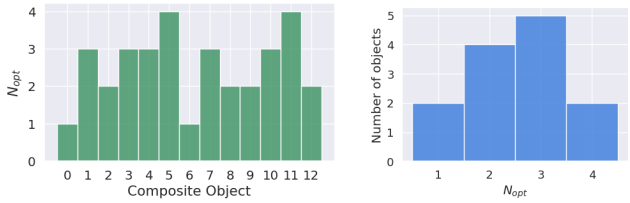
Using the features $F$ displayed in Fig. 3, we create a hand-crafted object set of $K = 13$ composite objects. These objects are defined by their abstract representations, which are listed in Tab. I. We manually select the features such that the objects are imbalanced with respect to the optimal policy, e.g., the number of glances under optimal exploration $N_{\text{opt}}$ has a high variance. We examine the distribution of $N_{\text{opt}}$ to assess the intrinsic difficulty (and imbalance) of our hand-crafted object set and establish a theoretical limit for any learned policy. Fig. 5a lists $N_{\text{opt}}$ for each object individually and Fig. 5b displays a histogram of $N_{\text{opt}}$ computed over all objects.

Using the optimal policy, we can observe that two objects can be identified within a single glance, while others require up to four glances – the theoretical maximum. This large variance in the number of required glances forces our learning framework to make use of dynamic exploration to approach the optimal policy. By averaging over $N_{\text{opt}}$ for all objects, we establish that the expected number of glances for optimal exploration is:

$$\mathbb{E}\left[N_{\text{opt}}\right] = 2.54$$

During learning, we set the maximum number of glances $N_{\text{max}}$ to 10 and choose a balancing factor $b$ of $1/4$. We apply Eq. 2 to obtain the resulting penalty $p = -0.05$. Based on these values, we can directly determine the expected episode reward an optimal policy receives by subtracting the expected total glance penalty from the positive classification reward:

$$\mathbb{E}\left[R_{\text{opt}}\right] = r_{c+} - p \cdot \mathbb{E}\left[N_{\text{opt}}\right] = 0.873$$




(a) Object-specific $N_{\text{opt}}$ (b) $N_{\text{opt}}$ histogram

Fig. 5: Distribution of $N_{\text{opt}}$ on the composite object set under an optimal policy. The object-specific $N_{\text{opt}}$ is shown in (a). The histogram of $N_{\text{opt}}$ over the entire object set is shown in (b).

| Step | Prob | Knowledge | Count | Glance Location | | | |
|---|---|---|---|---|---|---|---|
| | | | | 0 | 1 | 2 | 3 |
| t=0 | 100% | $(\_, \_, \_, \_)$ | 13 | 2.85 | 2.92 | 3.08 | **2.54** |
| t=1 | 8% | $(\_, \_, \_, 0)$ | 1 | 0.00 | 0.00 | 0.00 | - |
| | 8% | $(\_, \_, \_, 1)$ | 1 | 0.00 | 0.00 | 0.00 | - |
| | 15% | $(\_, \_, \_, 2)$ | 2 | 2.00 | **1.00** | 2.00 | - |
| | 31% | $(\_, \_, \_, 4)$ | 4 | **1.50** | 2.00 | 2.25 | - |
| | 38% | $(\_, \_, \_, 5)$ | 5 | **2.40** | 2.60 | **2.40** | - |

TABLE II: The exploration strategy of the optimal policy on our custom object set using abstract representations. Each feature we can observe is associated with a probability. The count refers to the number of possible objects. Depending on the observed feature, we select the glance location with the lowest expected number of remaining glances, highlighted in bold. A count of 1 indicates a non-ambiguous knowledge state; thus, the expected number of remaining glances is 0.

### C. The Optimal Policy

Using the tuple-based object representations, we can also represent the state of knowledge during exploration. For example, we represent the knowledge that feature 0 is at socket 1 and feature 5 is at socket 3 as $(\_, 0, \_, 5)$ and use the underscore as a placeholder for yet unknown features. Let's assume that, given this knowledge, the only two remaining objects are $(1, 0, 2, 5)$ and $(1, 0, 4, 5)$. In this simple case, it is apparent that an optimal policy can identify the object with one additional glance at socket 2. Thus, for this example, the expected number of remaining optimal glances $N_{\text{opt}}$ is:

$$\mathbb{E}_{(\_,0,\_,5)}\left[N_{\text{opt}}\right] = 1$$

Using dynamic programming, we can incrementally compute the optimal glancing location for any partial knowledge. This gives us an a priori expectation of the number of glances under an optimal policy that always classifies correctly: $\mathbb{E}_{(\_,\_,\_,\_)}\left[N_{\text{opt}}\right]$. With this analytical model, we postulate that our learned policy $\pi$ is optimal if its classification accuracy is 100% and the expected number of remaining glances $N_\pi$ at $t = 0$ is minimal:

$$\mathbb{E}_{t=0}\left[N_\pi\right] = \mathbb{E}_{(\_,\_,\_,\_)}\left[N_{\text{opt}}\right]$$

The optimal policy can be represented by a decision tree with the objects at its leaves and branches off at the nodes based on the features observed by the most informative glance at that state. This tree's average (remaining) depth represents the average number of remaining glances to perform for a perfect classification. Tab. II shows the first two levels of this tree ($t = 0$ and $t = 1$) for our object set, demonstrating that glance location 3 is the average optimal starting location with $2.54$ expected glances on average until an object can be uniquely identified.

### D. Model Implementation

We implement the network architectures for the composite object set based on the abstract framework presented in Sec. III-C. We set $d_{\text{hidden}}$ to 64. The embedding network consists of two sub-networks that encode the pressure $M$ and location $p, q$ separately. Starting with the $16\times16$ pressure image, we apply two convolutions with 8 filters, each followed by a $2 \times 2$ Max Pooling layer. This results in 8 feature maps of dimension $4\times4$, which are flattened to a vector of size 128 and further compressed to a final pressure embedding of size 32 using a two-layer MLP. The 3-dimensional position $p$ and 4-dimensional rotation $q$ are concatenated to a vector of size 7 and scaled up to a 32-dimensional embedding using a two-layer MLP. By concatenating both embeddings, we obtain a unified glance embedding $g$ of dimension $d_{\text{hidden}} = 64$.

We implement two different sequence models: an LSTM and a Transformer. The LSTM-based sequence model directly mimics HAM's approach. It comprises two stacked LSTM cells with a hidden dimension of 64.

The Transformer-based sequence model consists of two stacked encoder blocks with two attention heads. The fundamental principle of using the Transformer encoder is that each input token represents exactly one embedded haptic glance of dimension $d_{\text{hidden}}$. In addition to the sequence of embedded glances, we input an extra learnable token $l$ of the same dimension. While passing through the Transformer-Encoder blocks, this token attends to the glance tokens and generates the latent sequence representation $h_t$ of dimension $d_{\text{hidden}}$. Similar to the concept of an additional classification token known from the Vision Transformer (ViT) [18], the outputs of the glance tokens from the final encoder block are omitted.

The classification, decision, glance, and value networks are each connected to the 64-dimensional output $h_t$ of the sequence model. They are two-layer MLPs with a hidden dimension of 32 and an output dimension corresponding to their specific task. We employ the ReLU activation function for all hidden layers.

### E. Experimental Protocol

We model the environment described in Sec. IV-A with an open-source simulation based on MuJoCo [19] with ROS support[1]. The tactile sensor is simulated by discretization of hydrostatic pressures computed by hydroelastic contacts in the form of a simulation plugin introduced by Leins et al. in [20] and [21]. We employ the same tactile resolution as [7] and record a pressure pattern $M \in \mathbb{R}^{16\times16}$ using the simulated Myrmex sensor.

To avoid costly simulations during training, analogous to the experimental protocol of HAM [7], we pre-record glances in advance and construct a lookup table that spans a uniform grid over the space of glance parameters. Having pre-computed this table, instead of simulating a glance with its continuous parameters, we select the closest discretized parameters on this grid and query the lookup table for the pre-recorded pressure and location. For this experiment, we choose a sampling resolution of 61x61.

For a comparison to the static exploration approach of [7] we train our proposed architecture with a fixed number of glances $N \in \{1, 2, 3, 4\}$ to assess the change in accuracy

[1] https://github.com/ubi-agni/mujoco_ros_pkgs

| N | Sequence Model | Accuracy | Avg. N | Avg. Reward |
|---|---|---|---|---|
| $N = 1$ | LSTM | 36.39% | 1 | - |
| | Transformer | 37.28% | 1 | - |
| $N = 2$ | LSTM | 59.15% | 2 | - |
| | Transformer | 57.99% | 2 | - |
| $N = 3$ | LSTM | 84.02% | 3 | - |
| | Transformer | 84.61% | 3 | - |
| $N = 4$ | LSTM | 100% | 4 | - |
| | Transformer | 100% | 4 | - |
| Dynamic | LSTM | 100% | 2.74 | 0.862 |
| | Transformer | 100% | **2.65** | **0.865** |
| Dynamic | Optimal | 100% | 2.54 | 0.873 |

TABLE III: Comparison of the performance between dynamic exploration and a fixed number of glances $N \in \{1, 2, 3, 4\}$, ablated by sequence model. Optimal denotes the average statistics for the pre-computed optimal policy. Note that due to the changes in the reward model for dynamic exploration, rewards for fixed-glance models have been omitted from the table for clarity.

and reward in relation to the number of glances. We then compare these models to our dynamic exploration approach. To ablate the impact of the chosen type of sequence model, we train each variant with an LSTM-based sequence model and Transformer-based sequence model, respectively.

As discussed in Sec. III-D, we generate 10000 random glance sequences of lengths $N \in \{1, \ldots, 8\}$ for each object $c \in \{1, \ldots, K\}$ to pre-train the weights of the classification network, the sequence model and the embedding network.

We perform 200 RL training epochs with 1000 episodes each. Training is repeated 5 times for each model. We report the average classification accuracy of the best epochs for all training runs.

## V. Results

The results of the experiments are listed in III. For a fixed number of glances, we observe that for $N = 1$ the accuracy lies at **36**% and **37**% respectively for the LSTM and Transformer models. This corresponds to an average of 5 out of the 13 composite objects being classified correctly. With more flexibility to gather contextual information with $N = 2$ and $N = 3$, accuracy increases to **58**% and **84**%, respectively, corresponding to 7.5 and 11 correctly classified objects out of 13. Finally, **100**% is only achieved for $N \geq 4$ as expected from theoretical results. The theoretical optimum computed in Sec. IV-C suggests that 100% accuracy can be reached with an *average* of 2.54 glances. Indeed, the dynamic exploration models both come close to that optimum with an average number of glances of 2.65 and 2.74 for the Transformer and LSTM, respectively. Overall, we notice that the Transformer has a small advantage over the LSTM sequence model with regard to the classification accuracies in the fixed number of glances case, as well as to the average amount of glances and average reward in the dynamic exploration case.

The evolution of accuracy and number of glances until classification performed with the dynamic exploration approach is depicted in Fig. 6. Noticeably, shortly after the training is started, the number of glances strongly increases

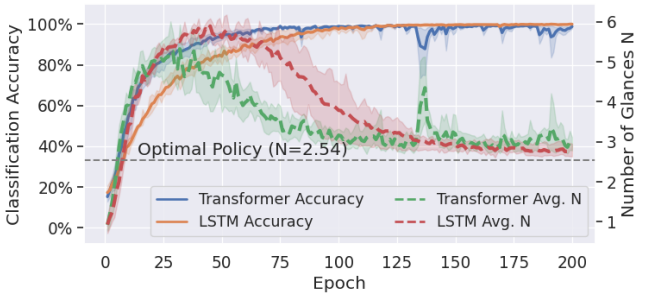



Fig. 6: Accuracy and average number of glances N over the course of training. Note that these results are obtained from probabilistic action generation during training and thus do not match the deterministic results reported in Tab. III

to a maximum. After about 50 epochs, it then begins to decay, slowly converging toward its minimum, which lies close to the optimum derived by the optimal policy. While the LSTM takes more time to reach the maximum amount of used glances and also converges slower, it behaves more stable than the Transformer variant. In terms of accuracy, a similar behavior can be seen: the Transformer reaches 100% accuracy much faster than the LSTM at about 60 epochs. However, unlike the LSTM, we see temporary decay in accuracy in later training epochs. We hypothesize that the transformer, due to its invariance to the ordering of glances in a sequence, profits more from the classification pre-training, resulting in faster rise in accuracy as well as decay in number of average glances. On the other hand, the increased stability of the LSTM might be a consequence of its lower complexity. Stronger discrepancies between both sequence models might arise when longer sequences and sensors with more complex morphologies are used, and the transformer is able to better harness its advantages over the LSTM.

Fig. 7 illustrates how the exploration strategy of the RL agent evolves over the course of the training. The exploration starts with a random distribution of glances, which grows until the slot locations of the features are reached. This corresponds to the phase in Fig. 6 where accuracy rises at the cost of average number of glances. Then the distribution splits into multiple local kernels covering the individual feature slots. This results in less uninformative glances, which entails reduction of number of average glances, while maintaining the classification accuracy. Noticeably, more important feature locations are indicated by the higher density at later stages of the training.

## VI. Conclusion

Performing exploration with the primitive objects of [7] showed that our approach performs better than HAM with even fewer glances. Furthermore, we find that the set of primitive shapes does not require multiple glances for accurate classification as the shapes are too easy to discriminate.

With the help of our custom-designed composite objects, we successfully validate the advantage to performing dynamic exploration on a highly unbalanced object set. We design a method to integrate the decision of when to obtain a glance into the model and associate glances with a cost.

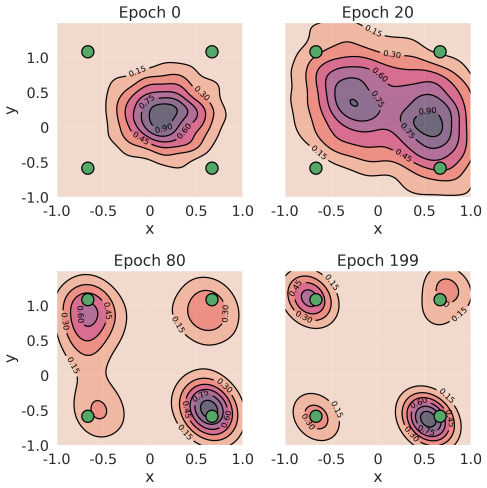



Fig. 7: Evolution of the exploration strategy as probability density function (PDF) normalized to a maximum density of 1. The PDF is estimated with Gaussian kernels on glances gathered from 1000 episodes sampled with the model (probabilistic) at the indicated epoch. The progression of the training shows a convergence toward the feature centers (green marks), starting from a centered random initialization.

With this conceptual change, both the Haptic Transformer, as well as the LSTM are able to learn an exploration that minimizes the number of glances while accurately classifying the entire object set. We clearly show that the number of glances can be substantially reduced by learning the decision of when to stop and remove the dependency on an additional hyperparameter.

Our composite objects, with features that are easily distinguishable on their own, possess a specific underlying structure and do not represent typical real-world objects. This synthetic environment is necessary to benchmark the results against a mathematically derivable optimum. However, the simplification of easily distinguishable features at specific locations does not reflect the shape and texture characteristics we would expect in a real-world scenario. Thus, with the findings of our work, a reasonable next step would be to apply our approach to a real-world task, like or object state estimation.

Furthermore, a sequence of glances could also be interpreted as a set of simultaneous glances performed by a dexterous manipulator. In that case, the kinematic constraints have to be additionally considered when choosing target glance locations to perform simultaneous multi-fingered exploration.

## References


[1] S. Luo, J. Bimbo, R. Dahiya, and H. Liu, "Robotic tactile perception of object properties: A review," *Mechatronics*, vol. 48, pp. 54–67, 2017.

[2] U. Martinez-Hernandez, A. Rubio-Solis, and T. J. Prescott, "Learning from sensory predictions for autonomous and adaptive exploration of object shape with a tactile robot," *Neurocomputing*, vol. 382, pp. 127–139, 2020.

[3] J. Fishel and G. Loeb, "Bayesian Exploration for Intelligent Identification of Textures," *Frontiers in Neurorobotics*, vol. 6, 2012.

[4] N. F. Lepora, U. Martinez-Hernandez, and T. J. Prescott, "Active touch for robust perception under position uncertainty," in *2013 IEEE International Conference on Robotics and Automation*. Karlsruhe, Germany: IEEE, May 2013, pp. 3020–3025.

[5] M. Kaboli, K. Yao, D. Feng, and G. Cheng, "Tactile-based active object discrimination and target object search in an unknown workspace," *Autonomous Robots*, vol. 43, pp. 123–152, 2019.

[6] S. Yang, W. D. Kim, H. Park, S. Min, H. Han, and J. Kim, "In-hand object classification and pose estimation with sim-to-real tactile transfer for robotic manipulation," *IEEE Robotics and Automation Letters*, vol. 9, no. 1, pp. 659–666, 2023.

[7] S. Fleer, A. Moringen, R. L. Klatzky, and H. Ritter, "Learning efficient haptic shape exploration with a rigid tactile sensor array," *PLOS ONE*, vol. 15, no. 1, pp. 1–22, 01 2020. [Online]. Available: https://doi.org/10.1371/journal.pone.0226880

[8] V. Mnih, N. Heess, and A. Graves, "Recurrent Models of Visual Attention."

[9] C. Schurmann, R. Kõiva, R. Haschke, and H. Ritter, "A modular high-speed tactile sensor for human manipulation research," in *2011 IEEE World Haptics Conference*, June 2011, pp. 339–344.

[10] R. L. Klatzky and S. J. Lederman, "Identifying objects from a haptic glance," *Perception & Psychophysics*, vol. 57, no. 8, pp. 1111–1123, Nov 1995. [Online]. Available: https://doi.org/10.3758/BF03208368

[11] R. J. Williams, "Simple statistical gradient-following algorithms for connectionist reinforcement learning," *Machine Learning*, vol. 8, no. 3, pp. 229–256, May 1992. [Online]. Available: https://doi.org/10.1007/BF00992696

[12] R. S. Sutton, D. McAllester, S. Singh, and Y. Mansour, "Policy gradient methods for reinforcement learning with function approximation," in *Advances in Neural Information Processing Systems*, S. Solla, T. Leen, and K. Müller, Eds., vol. 12. MIT Press, 1999. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/1999/file/464d828b85b0bed98e80ade0a5c43b0f-Paper.pdf

[13] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*, 2nd ed. The MIT Press, 2018. [Online]. Available: http://incompleteideas.net/book/the-book-2nd.html

[14] J. Schulman, P. Moritz, S. Levine, M. I. Jordan, and P. Abbeel, "High-dimensional continuous control using generalized advantage estimation," in *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*, Y. Bengio and Y. LeCun, Eds., 2016. [Online]. Available: http://arxiv.org/abs/1506.02438

[15] M. Neunert, A. Abdolmaleki, M. Wulfmeier, T. Lampe, J. T. Springenberg, R. Hafner, F. Romano, J. Buchli, N. Heess, and M. Riedmiller, "Continuous-discrete reinforcement learning for hybrid control in robotics," 2020.

[16] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal Policy Optimization Algorithms," Aug. 2017.

[17] J. Schulman, S. Levine, P. Moritz, M. I. Jordan, and P. Abbeel, "Trust Region Policy Optimization," Apr. 2017.

[18] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby, "An image is worth 16x16 words: Transformers for image recognition at scale," 2021.

[19] E. Todorov, T. Erez, and Y. Tassa, "Mujoco: A physics engine for model-based control," in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2012, pp. 5026–5033.

[20] D. Leins, F. Patzelt, and R. Haschke, "More Accurate Tactile Sensor Simulation with Hydroelastic Contacts in MuJoCo," in *IROS 2023 Workshop on Leveraging Models for Contact-Rich Manipulation*, Sept. 2023.

[21] F. Patzelt, D. Leins, and R. Haschke, "Curved Tactile Sensor Simulation with Hydroelastic Contacts in MuJoCo," *NeurIPS 2023 Workshop on Touch Processing: a new Sensing Modality for AI*, Dec. 2023.