Extreme Trust Region Policy Optimization for Active Object Recognition

Huaping Liu, Yupei Wu, and Fuchun Sun

***Abstract*— In this brief, we develop a deep reinforcement learning method to actively recognize objects by choosing a sequence of actions for an active camera that helps to discriminate between the objects. The method is realized using trust region policy optimization, in which the policy is realized by an extreme learning machine and, therefore, leads to efficient optimization algorithm. The experimental results on the publicly available data set show the advantages of the developed extreme trust region optimization method.**

***Index Terms*— Active object recognition, extreme learning machines (ELMs), trust region policy optimization (TRPO).**

I. Introduction

With the development of machine learning, especially the deep learning technology, the visual object recognition problem achieves great success in many domains. However, in many practical scenarios, only one single-view image is provided and the viewpoint ambiguity leads to great challenges for visual object recognition. If the observer is allowed to perceive the object from another view point when recognition fails from the first view point, it may be able to significantly improve the recognition performance. This can be achieved using the technology of active perception [1], which is a process of making sequences of decisions and has been extensively studied in various fields.

As a representative active perception technology, the active object recognition allows the camera to change its position or orientation to obtain more images about the object. This strategy significantly reduces the influence of the single-view ambiguity. During the past decade, many scholars developed a series of methods to tackle this problem. In [2], an Adaboost method was developed to exploit the relation among view points. In [3], a Bayesian inference model was proposed to integrate the processing of sensory and motor information. Andreopoulos and Tsotsos [4] derived some theoretical results for the problem of actively searching to determine the locations of prespecified objects. In [5], in-hand object recognition on the iCub humanoid robot was implemented. In addition, the active object recognition and pose estimation problems for household objects in a highly cluttered environment were addressed in [6]. An information theoretic framework that unified online feature selection and view planning was presented in [7]. A new active object detection and localization framework was developed in [8] to combine a robust untextured object detection and 3-D pose estimation algorithm.

Furthermore, the recent development of deep Q-network (DQN) to solve Atari games [9] inspired a lot of attentions from the fields of machine learning and robotics. Reference [10] developed a DQN for an active object recognition system, which learns to explore objects by minimizing classification error with short observation periods. Reference [11] established an active detection model to localize objects in an image. In [12], DQN was utilized for improved visual attention recognition. The main disadvantage of DQN is that it explores discrete action space. Therefore, the real-world application of DQN for active object recognition is still very challenging.

By randomly assigning hidden nodes on a simple single-hidden layer feed-forward neural networks, an extreme learning machine (ELM) was developed to tackle various tasks [13]–[15]. Not surprising, ELM provides strong supports for the reinforcement learning. For instance, the combination of ELM and Q function approximation was demonstrated in [16] to improve the learning efficiency. In [17], the reinforcement learning of single robot hose transport was performed using the ELM. The dynamics of linked multicomponent robotic systems encapsulated on a simulator using the ELM approach was learned in [18]. Reference [19] also showed some promising results using the ELM. However, the ELM has never been developed for image-based active object recognition tasks.

In this brief, we utilize the recently proposed trust region policy optimization (TRPO) framework [20] to tackle the active object recognition problem. The control policy is performed with an ELM network; therefore, more efficient optimization algorithms can be developed. We also show the advantages of the developed extreme trust region optimization method on publicly available data set.

The rest of this brief is organized as follows. In Section II, we give a brief review about the related work. Section III formulates the active object recognition problem. The proposed method is examined in Section IV and Section V presents the experimental validations.

II. Related Work

Several reinforcement learning methods had been proposed to solve the active object recognition problem and here we give a brief review. A reinforcement learning scheme has been proposed for exploration in [21]. Reference [22] presented a detailed survey on the reinforcement learning in robotics. For active viewpoint selection for object recognition, [23] attempted an unsupervised reinforcement learning algorithm for modeling of continuous states, continuous actions, sequential fusion of gathered image information, and supporting rewards for an optimized recognition. Reference [24] defined the recognition process as a sequential decision problem with the objective to disambiguate initial object hypotheses. Reinforcement learning provided an efficient approach to autonomously develop near-optimal decision strategies in terms of sensorimotor mappings. Reference [25] proposed to learn the movement of a visual sensor intentionally and to classify a word from the series of partial information simultaneously only based on the reward and punishment generated from the recognition result. Reference [26] presented a reinforcement learning method to decide which algorithm should be used to recognize objects seen by a mobile robot in an indoor environment. Reference [27] used reinforcement learning to explore the action space of the control problem. Reference [28] analyzed the influence of three different, intuitive, and optimized methods for handling invalid action suggestions generated by reinforcement learning.

Manuscript received April 3, 2017; revised October 18, 2017 and December 14, 2017; accepted December 15, 2017. Date of publication January 15, 2018; date of current version May 15, 2018. This work was supported in part by the National Natural Science Foundation of China under Grant U1613212, Grant 61673238, and Grant 61327809, in part by the National High-Tech Research and Development Plan under Grant 2015AA042306, and in part by the Beijing Municipal Science and Technology Commission under Grant D171100005017002. *(Corresponding author: Huaping Liu.)*

The authors are with the State Key Laboratory of Intelligent Technology and Systems, Department of Computer Science and Technology, Tsinghua University, Beijing 100084, China (e-mail: hpliu@tsinghua.edu.cn).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TNNLS.2017.2785233

Despite the successful applications of the reinforcement learning in active perception, the deep reinforcement learning has just been developed for such tasks [10]. However, in [10], only the discrete action space was explored. In this brief, we exploit an efficient policy learning method for continuous action exploration to tackle the active object recognition problem.

III. Problem Formulation

During active recognition step, an agent movement is performed to a new viewing position at which a new image is captured. The viewing position is known to the agent. Concretely speaking, at the initialization instant $t = 0$, the pose of the camera is $\boldsymbol{o}_0$ and the captured image is $I_{\boldsymbol{o}_0}$. We can then take $I_{\boldsymbol{o}_0}$ as the testing sample to get the label. However, since the single view data may not sufficient to give robust results, we should resort another view to improve the classification capability. This requires us to develop an action $\boldsymbol{a}_t$ to get

$$\boldsymbol{o}_{t+1} = \boldsymbol{o}_t + \boldsymbol{a}_t.$$

Then the camera is driven to get the new pose $\boldsymbol{o}_{t+1}$ and take $I_{\boldsymbol{o}_{t+1}}$ as the next input. Such a procedure is repeated for T steps and the obtained image set $I_{\boldsymbol{o}_0}, I_{\boldsymbol{o}_1}, \ldots, I_{\boldsymbol{o}_T}$ is all used for classification for this object.

In this brief, we regard such a problem as a reinforcement learning one. To this end, we should use some predeveloped training instances to learn the action selection strategy. Here, one training instance means a complete image sequence that contains multiple different views of one object. For the nth instance, the image sample that is captured at pose $\boldsymbol{o}$ is denoted as $I_{\boldsymbol{o}}^{(n)}$.

In a reinforcement learning setting, a learning agent interacts with an unknown environment in order to achieve a goal. In this setting, the goal is to maximize the cumulative reward accumulated by adapting its behavior within the environment. To this end, we introduce the following symbols.

1) $\mathcal{S}$ is the set of continuous states, which are produced by the acquired images.
2) $\mathcal{A}$ is the set of continuous actions, which are determined by the adopted agent.
3) $\mathcal{R} : \mathcal{S} \to \mathbb{R}$ is the reward function.
4) $\mathcal{P} : \mathcal{S} \times \mathcal{S} \times \mathcal{A} \to \mathcal{S}$ is the state transition probability, which describes the effect of each action in each state.

At each time step t, the agent receives the state $\boldsymbol{s}_t \in \mathcal{S}$, executes the action $\boldsymbol{a}_t \in \mathcal{A}$, produces the next state $\boldsymbol{s}_{t+1} \sim \mathcal{P}(\boldsymbol{s}_{t+1}|\boldsymbol{s}_t, \boldsymbol{a}_t) \in \mathcal{S}$, and obtains the scalar reward $r_t = \mathcal{R}(\boldsymbol{s}_t)$.

IV. Proposed Method

The goal of reinforcement learning is to learn a policy $\pi : \mathcal{S} \times \mathcal{A} \to [0, 1]$, which maximizes the expected return over all episodes. Under a stationary policy π, we use the Q-value function for selecting action $\boldsymbol{a}$ in state $\boldsymbol{s}$ to represent the expected future discounted return

$$Q_\pi(\boldsymbol{s}, a) = \mathop{\mathbb{E}}_{\substack{\boldsymbol{s}_{t+1} \sim \mathcal{P}(\boldsymbol{s}_{t+1}|\boldsymbol{s}_t, \boldsymbol{a}_t) \\ \boldsymbol{a}_{t+1} \sim \pi(\boldsymbol{a}_{t+1}|\boldsymbol{s}_{t+1})}} \left[\sum_{t=0}^{T} \{\gamma^t r_t | \boldsymbol{s}_0 = s, \boldsymbol{a}_0 = a\} \right] \tag{1}$$

where T is the length of the concerned time domain, which may be infinity, and $\gamma \in (0, 1]$ is a discount factor for future rewards. This value function is a measure of the long-term reward obtained by taking action $\boldsymbol{a}$ at state $\boldsymbol{s}$.

In addition, we define $\rho_0 : \mathcal{S} \to R$ as the distribution of the initial state $\boldsymbol{s}_0$, and the unnormalized discounted visitation frequencies ρ_π as

$$\rho_\pi = \sum_{t=0}^{T} \gamma^t P(\boldsymbol{s}_t = \boldsymbol{s})$$

where $\boldsymbol{s}_0 \sim \rho_0$ and the actions are selected according to π.

We assume that the policy π is parameterized by $\boldsymbol{\theta}$ and denote it as π_θ. The core task is then to iteratively update the parameter $\boldsymbol{\theta}$ to improve the policy to accomplish the final goal. We, therefore, resort the recently developed TRPO framework for such a task.

Let $\pi_{\theta_{\text{old}}}$ be the old policy and π_θ be the new policy after the policy update. Using this notation, we can formulate the following constrained optimization problem:

$$\begin{aligned} &\max_{\boldsymbol{\theta}} \ \mathbb{E}_{\boldsymbol{s} \sim \rho_{\theta_{\text{old}}}, \boldsymbol{a} \sim q} \left[\frac{\pi_\theta(\boldsymbol{a}|\boldsymbol{s})}{q(\boldsymbol{a}|\boldsymbol{s})} Q_{\pi_{\theta_{\text{old}}}}(\boldsymbol{s}, \boldsymbol{a}) \right] \\ &\text{s.t. } \mathbb{E}_{\boldsymbol{s} \sim \rho_{\theta_{\text{old}}}} [D_{KL}(\pi_{\theta_{\text{old}}}(\cdot|\boldsymbol{s}) || \pi_\theta(\cdot|\boldsymbol{s}))] \le \delta \end{aligned} \tag{2}$$

where $D_{KL}(\cdot, \cdot)$ is the KL-divergence between two distributions and the parameter δ is used to bound the KL-divergency between the two policies. $q(\boldsymbol{a}|\boldsymbol{s})$ is a distribution which is easy to sample the actions and it is natural to set it as

$$q(\boldsymbol{a}|\boldsymbol{s}) = \pi_{\boldsymbol{\theta}_{\text{old}}}(\boldsymbol{a}|\boldsymbol{s}).$$

In this case, we collect a sequence of states by sampling $\boldsymbol{s}_0 \sim \rho_0$ and then simulating the policy $\pi_{\theta_{\text{old}}}$ for some number of time steps to generate a trajectory. $Q_{\theta_{\text{old}}}(\boldsymbol{s}, \boldsymbol{a})$ is computed at each state-action pair $(\boldsymbol{s}_t, \boldsymbol{a}_t)$ by taking the discounted sum of future costs along the trajectory.

In the following, we will introduce how to prepare samples for establishing optimization problem in (2) and develop efficient optimization algorithm using an efficient neural network.

A. ELM for Policy Network

Since the policy is parameterized by $\boldsymbol{\theta}$, how to obtain $\boldsymbol{\theta}$ becomes the core of the problem. In previous work, a multilayer neural network is usually used to represent the policy. This leads to huge optimization parameters, which reduces the times efficacy and increases the optimization difficulties [29]. To solve this problem, we use the ELM with fixed random weights to represent the policy to reduce the calculation.

The basic idea of ELM is to develop a neural network with random weights to get new feature representations. According to the core idea of ELM, the hidden layer activation function is denoted as

$$h_l(\boldsymbol{s}) = \frac{1}{1 + \exp\big(-\big(\boldsymbol{w}_l^T \boldsymbol{s} + b_l\big)\big)} \quad \text{for } l = 1, 2, \ldots, L$$

where L is the number of hidden nodes. The parameters $\boldsymbol{w}_l \in \mathbb{R}^L$ and $b_l \in \mathbb{R}$ are randomly determined according to specified probability distribution.

Therefore, ELM transforms the state vector $\boldsymbol{s}$ as the new feature vector

$$\boldsymbol{h}(\boldsymbol{s}) = [h_1(\boldsymbol{s}) \ h_2(\boldsymbol{s}) \ldots h_L(\boldsymbol{s})]^T \in R^L.$$

The action output node of ELM is calculated as

$$\boldsymbol{o} = \boldsymbol{\theta}^T \boldsymbol{h}(\boldsymbol{s})$$

where $\boldsymbol{\theta}$ is the output weight, which should be determined.

In this brief, the policy $\pi_\theta(\boldsymbol{a}|\boldsymbol{s})$ is developed as a Gaussian distribution, which is represented as

$$\pi_\theta(\boldsymbol{a}|\boldsymbol{s}) = \mathcal{N}(\boldsymbol{\theta}^T \boldsymbol{h}(\boldsymbol{s}), \Sigma)$$

where Σ is the prescribed covariance matrix, which is not involved in the optimization procedure.

B. E-TRPO Algorithm

In this section, we introduce how to determine the parameter $\boldsymbol{\theta}$, which helps mapping from the visual features vector $\boldsymbol{s}$ to the pose adjustments $\boldsymbol{a}$. The whole algorithm is listed in Algorithm 1, which can be divided into two parts.

First, we should replace the expectations by sample averages and replace the Q value by an empirical estimate. In this way, we collect a sequence of states by sampling $s_0 \sim \rho_0$ and then simulating the policy $\pi_{\theta_{\text{old}}}$ for some number of time steps to generate a trajectory

$$s_0, \boldsymbol{a}_0, s_1, \boldsymbol{a}_1, \ldots, s_{T-1}, \boldsymbol{a}_{T-1}, s_T.$$

Then $Q_{\pi_{\theta_{\text{old}}}}(\boldsymbol{s}, \boldsymbol{a})$ can be computed at each state-action pair $(\boldsymbol{s}_t, \boldsymbol{a}_t)$ by taking the discounted cum of future costs along the trajectory. This forms the first part (Lines 5 and 6) of the whole algorithm.

After that we can approximate the optimization problem in (2) as

$$\max_{\boldsymbol{\theta}} \ \mathcal{L}_\theta = \frac{1}{T+1}\sum_{t=0}^{T} \frac{\pi_\theta(\boldsymbol{a}_t|\boldsymbol{s}_t)}{\pi_{\text{old}}(\boldsymbol{a}_t|\boldsymbol{s}_t)} Q_{\pi_{\theta_{\text{old}}}}(\boldsymbol{s}_t, \boldsymbol{a}_t)$$

$$\text{s.t. } \mathcal{D}_\theta = \frac{1}{T+1}\sum_{t=0}^{T}\left[D_{KL}(\pi_{\theta_{\text{old}}}(\cdot|\boldsymbol{s}_t)||\pi_\theta(\cdot|\boldsymbol{s}_t))\right] \le \delta. \quad (3)$$

Algorithm 1 Reinforcement Learning for Active Object Recognition

Input: Training sequences $\{\mathcal{V}_n\}_{n=1}^N$.
Output: Parameter $\boldsymbol{\theta}$.

```
1: Initialize $\boldsymbol{\theta}$ as a random vector.
2: for episode = 1 to $M$ do
3:   for $n = 1$ to $N$ do
4:     $\boldsymbol{\theta}_{\text{old}} = \boldsymbol{\theta}$
5:     For sequence $\mathcal{V}_n$, collect a path
           $\{\boldsymbol{s}_t, \boldsymbol{a}_t, r_t, \boldsymbol{s}_{t+1}\}$
       for $t = 0, 2, \ldots, T-1$, according to $\pi_{\theta_{\text{old}}}(\boldsymbol{a}|\boldsymbol{s})$. Please note
       that $\boldsymbol{s}_0$ is extracted from the first frame of $\mathcal{V}_n$ and
           $\boldsymbol{a}_t \sim \pi_{\theta_{\text{old}}}(\boldsymbol{a}|\boldsymbol{s} = \boldsymbol{s}_t),$
                                                              (4)
           $\boldsymbol{s}_{t+1} \sim \mathcal{P}(\boldsymbol{s}_{t+1}|\boldsymbol{s}_t, \boldsymbol{a}_t)$
6:     For each $t$, calculate $Q_{\pi_{\theta_{\text{old}}}}(\boldsymbol{s}_t, \boldsymbol{a}_t)$ according to
           $Q_{\pi_{\theta_{\text{old}}}}(\boldsymbol{s}_t, \boldsymbol{a}_t) = \begin{cases} r_T & t = T \\ r_t + \gamma Q_{\pi_{\theta_{\text{old}}}}(\boldsymbol{s}_{t+1}, \boldsymbol{a}_{t+1}) & t < T \end{cases}$
7:     Initialize $L_n^{(0)} = \frac{1}{T+1}\sum_{t=0}^{T} Q_{\pi_{\theta_{\text{old}}}}(\boldsymbol{s}_t, \boldsymbol{a}_t)$
8:     Calculate the maximum step size $\beta_{max}$ according to Eq. (7).
9:     while 1 do
10:      Set $\beta^{(k)} = 0.5^{k-1}\beta_{max}$
11:      $\boldsymbol{\theta}^{(k)} = \boldsymbol{\theta}_{\text{old}} + \beta^{(k)}\boldsymbol{s}_{\theta_{\text{old}}}$
12:      $L_n^{(k)} = \frac{1}{T+1}\sum_{t=0}^{T}\frac{\pi_{\theta^{(k)}}(\boldsymbol{a}_t|\boldsymbol{s}_t)}{\pi_{\theta_{\text{old}}}(\boldsymbol{a}_t|\boldsymbol{s}_t)} Q_{\pi_{\theta_{\text{old}}}}(\boldsymbol{s}_t, \boldsymbol{a}_t)$
13:      if $L_n^{(k)} > L_n^{(k-1)}$ then
14:        break
15:      end if
16:    end while
17:    $\boldsymbol{\theta} = \boldsymbol{\theta}^{(k)}$
18:  end for
19: end for
20: OUTPUT: $\boldsymbol{\theta}$.
```

To solve (3), we should first calculate the gradient as

$$\boldsymbol{g}_{\theta_{\text{old}}} = \frac{\partial \mathcal{L}_\theta}{\partial \boldsymbol{\theta}}\Big|_{\boldsymbol{\theta}=\boldsymbol{\theta}_{\text{old}}} = \frac{1}{T+1}\sum_{t=0}^{T}\frac{Q_{\text{old}}(\boldsymbol{s}_t, \boldsymbol{a}_t)}{\pi_{\text{old}}(\boldsymbol{a}_t|\boldsymbol{s}_t)}\frac{\partial \pi_\theta(\boldsymbol{a}_t|\boldsymbol{s}_t)}{\partial \boldsymbol{\theta}}\Big|_{\boldsymbol{\theta}=\boldsymbol{\theta}_{\text{old}}}. \quad (5)$$

Furthermore, by approximating

$$\mathcal{D}_\theta \doteq \frac{1}{2}(\boldsymbol{\theta} - \boldsymbol{\theta}_{\text{old}})^T \boldsymbol{F}(\boldsymbol{\theta} - \boldsymbol{\theta}_{\text{old}}) \quad (6)$$

where $\boldsymbol{F}$ is the Fisher information matrix, of which (i, j)th element is $(\partial^2/\partial\theta_i\partial\theta_j)\mathcal{D}_{\boldsymbol{\theta}}$, and we can determine the search direction according to the natural gradient

$$\boldsymbol{s}_{\theta_{\text{old}}} = \boldsymbol{F}^{-1}\boldsymbol{g}_{\theta_{\text{old}}}.$$

The last thing is to determine the search step β, which is used to obtain the updated solution

$$\boldsymbol{\theta} = \boldsymbol{\theta}_{\text{old}} + \beta \boldsymbol{s}_{\theta_{\text{old}}}.$$

According to (6), the constraint condition in (3) can be represented as

$$\frac{1}{2}\beta^2 \boldsymbol{s}_{\theta_{\text{old}}}^T \boldsymbol{F}\boldsymbol{s}_{\theta_{\text{old}}} \le \delta$$

and, therefore, the maximum step size can be determined as

$$\beta_{max} = \sqrt{\frac{2\delta}{\boldsymbol{s}_{\theta_{\text{old}}}^T \boldsymbol{F}\boldsymbol{s}_{\theta_{\text{old}}}}}. \quad (7)$$

Even though the maximum step size can be achieved, in practical implementation, we usually use an extra line search stage to get more stable results. This forms Lines 9–16 of Algorithm 1.

Remark 1: If π takes the univariate Gaussian form, which can be represented as

$$\pi_\theta(a_t|\boldsymbol{s}_t) = \exp\left(-\frac{(a_t - \boldsymbol{\theta}^T\boldsymbol{h}(\boldsymbol{s}_t))^2}{2\sigma^2}\right)$$

where σ is the variance, we can calculate

$$\frac{\partial \pi_\theta(a_t|\boldsymbol{s}_t)}{\partial \boldsymbol{\theta}} = e^{\left(-\frac{(a_t - \boldsymbol{\theta}^T\boldsymbol{h}(\boldsymbol{s}_t))^2}{2\sigma^2}\right)}\left(-\frac{a_t - \boldsymbol{\theta}^T\boldsymbol{h}(\boldsymbol{s}_t)}{\sigma^2}\boldsymbol{h}(\boldsymbol{s}_t)\right).$$

On the other hand, $\mathcal{D}_\theta$ can be represented as

$$\mathcal{D}_\theta = \frac{1}{(T+1)\sigma}\sum_{t=0}^{T}\left(\boldsymbol{\theta}^T\boldsymbol{h}(\boldsymbol{s}_t) - \boldsymbol{\theta}_{\text{old}}^T\boldsymbol{h}(\boldsymbol{s}_t)\right)^T\left(\boldsymbol{\theta}^T\boldsymbol{h}(\boldsymbol{s}_t) - \boldsymbol{\theta}_{\text{old}}^T\boldsymbol{h}(\boldsymbol{s}_t)\right)$$

and, therefore, the (i, j)th element of the Fisher information matrix can be calculated as

$$F_{ij} = \frac{\partial \mathcal{D}_\theta}{\partial\theta_i\theta_j} = \frac{2}{T\sigma^2}\sum_{t=0}^{T} h_i(\boldsymbol{s}_t)h_j(\boldsymbol{s}_t)$$

where $h_i(\boldsymbol{s}_t)$ and $h_j(\boldsymbol{s}_t)$ are the ith and jth element of the vectors $\boldsymbol{h}(\boldsymbol{s}_t)$, respectively.

Remark 2: The constraint condition presented in (3) is used to control the size of the policy update to get a reasonable step size. This is the core motivation of TRPO algorithm, which has been proved to exhibit monotonic improvement property [20]. Nevertheless, such algorithms can only achieve local optimum. This is a common problem, which occurs in many reinforcement learning algorithms. In practice, we can adopt some heuristic methods to improve the performance, such as multiple initializations.

C. Reward Function

Similar to [10], we transform the image into the CNN feature by adding a softmax layer top of the CNN model to recognize the concerned objects. The output of this softmax layer is denoted as $P(c|I)$, where I is the given image and c is the object label. $P(c|I)$ serves as the belief over different objects given an image. This belief is combined with the accumulated belief from the previous images using Naive Bayes. That is to say, we calculate

$$P(c|I_{o_0}, I_{o_1}, \ldots, I_{o_t}) \propto \prod_{l=0}^{t} P(c|I_{o_l}).$$

The obtained belief $P(c|I_{o_0}, I_{o_1}, \ldots, I_{o_t})$ can be used for recognizing object according to

$$\hat{y}_t = \text{argmax}_c P(c|I_{o_0}, I_{o_1}, \ldots, I_{o_t}).$$

Obviously, the concentration of the belief reflects the confidence of CNN to the classification of the input images. The more flat the probability distribution of a classification is, the more difficult it is to distinguish the input images. This inspires us to use information entropy to quantitatively measure the amount of the uncertainty [12]. We denote $H(I_{o_0:o_t})$ as the information entropy of the classification probability $P(c|I_{o_0}, I_{o_1}, \ldots, I_{o_t})$, and use it as the evaluation metric to judge the discrimination of the selected image and the classification confidence. A small information entropy means it is easy to discriminate the image with more confidence, which can guide us to find the key images of the image to the classification.

If the predicted result of $I_{o_0:o_t}$ is wrong, the rewards should be definitely negative. On the other hand, if the predicted result of $I_{o_0:o_t}$ is right, we should consider two different cases. When the information entropy $H(I_{o_0:o_t})$ is smaller than $H(I_{o_0:o_{t-1}})$, it means that the new selected image is useful for our task. Then, the agent would receive a positive reward. On the contrary, the agent receives zero reward. The complete rule for the reward is, therefore, developed as

$$r_t = \begin{cases} -1 & \hat{y}_t \neq y^* \\ 0 & \hat{y}_t = y^* \text{ AND } H(I_{o_0:o_t}) > H(I_{o_0:o_{t-1}}) \\ +1 & \hat{y}_t = y^* \text{ AND } H(I_{o_0:o_t}) \leq H(I_{o_0:o_{t-1}}) \end{cases}$$

where y^* is the true label of the object. Such a reward evaluates the quality of an action in a given state and provides the information about the goal of the reinforcement learning agent.

D. Active Object Recognition

Once the optimal solution $\boldsymbol{\theta}^*$ is obtained, we can use it for the practical active object recognition task. A subtle difference lies in that the action is obtained as $\boldsymbol{a}_t = \boldsymbol{\theta}^{*T}\boldsymbol{h}(\boldsymbol{s}_t)$, where $\boldsymbol{s}_t$ is the state vector at time instance t. Then, the next pose is obtained as $\boldsymbol{o}_{t+1} = \boldsymbol{o}_t + \boldsymbol{a}_t$ and the agent adjusts the pose to obtain the new image. Similar to the training stage, we accumulate the belief and use it to determine the class label of the object. In some practical scenarios, they may not exhibit image corresponding to the exact pose of $\boldsymbol{o}_{t+1}$, and we just search the nearest available pose.

V. Experimental Results

A. Data Set

The adopted data set for performance evaluation is GERMS [10], which has 1365 videos recordings of give-and-take trials using 136 different object instances. The object instances are soft toys representing different human cell types, microbes, and disease-related organisms. In Fig. 1, we show some visual samples for all of the objects. Each video records the robot bringing the grasped object to its center of view, rotating it by 180° and then returning it. All of the video clips were recorded from robot's head-mounted camera at 30 frames/s, thus providing us the opportunities to use continuous-action-based policy search methods. We use the train-test subsets specified by the data set authors. The training data consist of six video clips per object, for a total of 816 clips.

Fig. 1. GERMS data set [10].

TABLE I
Parameter Setting

Parameter	Meaning	Value
M	The number of the episodes	20
γ	The discount factor	1
σ	The variance of the Gaussian distribution	0.01
δ	The bound of the KL divergency	0.01
L	The number of the hidden nodes	1000

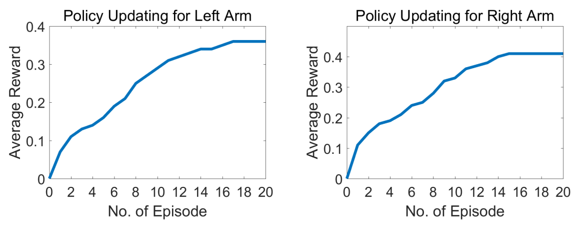

Fig. 2. Average reward versus the episode.

B. Implementation

Here, we provide some implementation details of the proposed E-TRPO algorithm. The GERMS data set contains two main tasks: the left arm and the right arm task. Each image frame is represented by a 4096-D VGG-net feature vector, provided in [10]. The active object recognition for GERMS exhibits a single degree of motion; therefore, the policy π takes the form which is provided in Remark 1. The parameters used in E-TRPO are listed in Table I. Please note that we simply set the discount factor as 1 and the parameters σ and δ are empirically set according to the original implementation of TRPO. In Fig. 2, we show the averaged reward values after each episode for $L = 1000$. In both the cases, the agent is able to improve his behaviors significantly. The policy starts at the average reward 0.1 or so and converges to about 0.4 after 20 episodes; therefore, we set $M = 20$. In addition, we set the maximum step T for recognition as $T = 10$.

C. Result Comparison

To evaluate the proposed active object recognition performance, we test the following methods on the test set of GERMS data set.

1) *Sequential Strategy:* It always starts from the same position and moves in the same direction to the next immediate position.

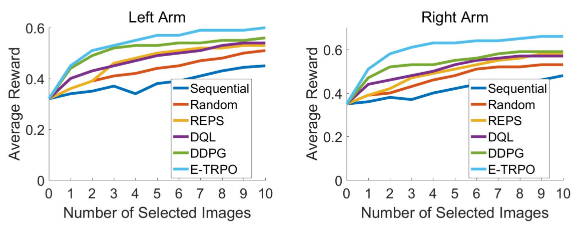

Fig. 3. Performance comparison among different methods. The results show that the proposed method could achieve the best results.

2) *Random Strategy:* It selects a random action with uniform probability. Due to the random nature, we repeatedly perform this strategy for 20 times and report the average results.
3) *DQL Strategy:* It was developed in [10] using deep Q-learning method. Since DQL deals with discrete action space only, we have to choose the rotation offsets from the action set $\{\pm\pi/64, \pm\pi/32, \pm\pi/16, \pm\pi/8, \pm\pi/4\}$.
4) *REPS Strategy:* It was originally developed in [30] by using a relative entropy policy search (REPS) method. For fair comparison, we replace the TRPO search module with REPS policy search module and the other modules are kept unchanged.
5) *DDPG Strategy:* It was developed in [31] to continuously improve the policy using gradient descent to the policy with minibatch data sampled from a replay pool.

Among the above methods, the sequential and random ones serve as the baselines. Both DQL and deep deterministic policy gradient (DDPG) are popular deep reinforcement learning and represent the state-of-the-art results. REPS, which constrains the state action marginal, is a representative of the policy search methods.

The performance is evaluated using the recognition accuracy, which is averaged over the entire set of images in the test set. Fig. 3 shows the accuracy function of the number of selected images for the two arms. Note that the case when the number is zero corresponds to recognizing the objects using the first frame of the test video instance. From the results, we make the following observations.

1) The performance of sequential strategy is rather poor, because it always select the next image which does not provide sufficient new information for recognition.
2) The proposed method performs consistently than the other ones. Especially, we find that it shows significant advantages for smaller steps. This is due to the adopted continuous-action space which is more robust and flexible.
3) The difference between REPS and the proposed E-TRPO shows the merits of the objective function adopted by the TRPO method. Another possible reason is that the complicated nonlinear optimization produced in the inner loop of REPS prevents the algorithm from searching better solutions.
4) The performance of DDPG is similar to that of E-TRPO for small steps. However, when the step increases, the difference becomes significant. This illustrates that E-TRPO selects more effective images for recognition.

VI. Conclusion

The problem of active object recognition intrinsically requires exploring the continuous action space. The DQN method, which deals with discrete action space, has to sample the action space and may introduce significant quantization error. Therefore, we resort to the framework of TRPO, which explores the continuous action space, to tackle this problem. Specifically, we develop an extreme TRPO method for active object recognition. The ELM is used for the policy network to effectively simplify the design procedure. The experimental results on the publicly available data set show the advantages of the developed extreme trust region optimization method.

References

[1] S. Chen, Y. Li, and N. M. Kwok, "Active vision in robotic systems: A survey of recent developments," *Int. J. Robot. Res.*, vol. 30, no. 11, pp. 1343–1377, 2011.

[2] Z. Jia, Y.-J. Chang, and T. Chen, "Active view selection for object and pose recognition," in *Proc. ICCVW*, Sep./Oct. 2009, pp. 641–648.

[3] D. Nakath, T. Kluth, T. Reineking, C. Zetzsche, and K. Schill, "Active sensorimotor object recognition in three-dimensional space," in *Proc. Int. Conf. Spatial Cognit.*, 2014, pp. 312–324.

[4] A. Andreopoulos and J. K. Tsotsos, "A computational learning theory of active object recognition under uncertainty," *Int. J. Comput. Vis.*, vol. 101, no. 1, pp. 95–142, 2013.

[5] B. Browatzki, V. Tikhanoff, G. Metta, H. H. Bülthoff, and C. Wallraven, "Active in-hand object recognition on a humanoid robot," *IEEE Trans. Robot.*, vol. 30, no. 5, pp. 1260–1269, Oct. 2014.

[6] K. Wu, R. Ranasinghe, and G. Dissanayake, "Active recognition and pose estimation of household objects in clutter," in *Proc. ICRA*, May 2015, pp. 4230–4237.

[7] C. Potthast, A. Breitenmoser, F. Sha, and G. S. Sukhatme, "Active multi-view object recognition: A unifying view on Online feature selection and view planning," *Robot. Auto. Syst.*, vol. 84, pp. 31–47, Oct. 2016.

[8] M. Imperoli and A. Pretto, "Active detection and localization of textureless objects in cluttered environments," in *Proc. CVIU*, 2016, pp. 1–18.

[9] V. Mnih *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, pp. 529–533, Feb. 2015.

[10] M. Malmir, K. Sikka, D. Forster, J. Movellan, and G. W. Cottrell, "Deep Q-learning for active recognition of germs: Baseline performance on a standardized dataset for active learning," in *Proc. BMVC*, 2015, pp. 1–32.

[11] J. C. Caicedo and S. Lazebnik, "Active object localization with deep reinforcement learning," in *Proc. ICCV*, 2015, pp. 2488–2496.

[12] D. Zhao, Y. Chen, and L. Lv, "Deep reinforcement learning with visual attention for vehicle classification," *IEEE Trans. Cogn. Develop. Syst.*, vol. 9, no. 4, pp. 356–367, Dec. 2017.

[13] G.-B. Huang, Q.-Y. Zhu, and C.-K. Siew, "Extreme learning machine: Theory and applications," *Neurocomputing*, vol. 70, nos.1–3, pp. 489–501. 2006.

[14] G.-B. Huang, H. Zhou, X. Ding, and R. Zhang, "Extreme learning machine for regression and multiclass classification," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 42, no. 2, pp. 513–529, Apr. 2012.

[15] H. Liu, J. Qin, F. Sun, and D. Guo, "Extreme kernel sparse learning for tactile object recognition," *IEEE Trans. Cybern.*, vol. 47, no. 12, pp. 4509–4520, Dec. 2017.

[16] J. Pan, X. Wang, Y. Cheng, and G. Cao, "Reinforcement learning based on extreme learning machine," in *Proc. Int. Conf. Intell. Comput.*, 2012, pp. 80–86.

[17] J. M. Lopez-Guede, B. Fernandez-Gauna, and M. Grana, "State-action value function modeled by ELM in reinforcement learning for hose control problems," *Int. J. Uncertainty, Fuzziness Knowl.-Based Syst.*, vol. 21, pp. 99–116, Dec. 2013.

[18] J. M. Lopez-Guede, B. Fernandez-Gauna, and J. A. Ramos-Hernanz, "A L-MCRS dynamics approximation by ELM for Reinforcement Learning," *Neurocomputing*, vol. 150, pp. 116–123, Feb. 2015.

[19] J. Hwangbo, C. Gehring, D. Bellicoso, P. Fankhauser, R. Siegwart, and M. Hutter, "Direct state-to-action mapping for high DOF robots using ELM," in *Proc. IROS*, Sep./Oct. 2015, pp. 2842–2847.

[20] J. Schulman, S. Levine, P. Moritz, M. I. Jordan, and P. Abbeel. (2015). "Trust region policy optimization." [Online]. Available: https://arxiv.org/abs/1502.05477

[21] T. Kollar and N. Roy, "Trajectory optimization using reinforcement learning for map exploration," *Int. J. Robot. Res.*, vol. 27, no. 2, pp. 175–196, 2008.

[22] J. Kober, J. A. Bagnell, and J. Peters, "Reinforcement learning in robotics: A survey," *Int. J. Robot. Res.*, vol. 32, no. 11, pp. 1238–1274, 2013.

[23] F. Deinzer, C. Derichs, H. Niemann, and J. Denzler, "A framework for actively selecting viewpoints in object recognition," *Int. J. Pattern Recognit. Artif. Intell.*, vol. 23, no. 4, pp. 765–799, 2009.

[24] L. Paletta and A. Pinz, "Active object recognition by view integration and reinforcement learning," *Robot. Auto. Syst.*, vol. 31, nos. 1–2, pp. 71–86, 2000.

[25] A. A. M. Faudzi and K. Shibata, "Acquisition of context-based active word recognition by Q-learning using a recurrent neural network," in *Robot Intelligent Technology and Applications 2*. Berlin, Germany: Springer, 2014, pp. 191–200.

[26] R. A. C. Bianchi, A. Ramisa, and R. L. de Mantaras, "Automatic selection of object recognition methods using reinforcement learning," in *Advances in Machine Learning I*. Berlin, Germany: Springer, 2010, pp. 421–439.

[27] A. D. Bagdanov, A. D. Bimbo, and F. Pernici, "Acquisition of high-resolution images through on-line saccade sequence planning," in *Proc. VSSN*, 2005, pp. 121–130.

[28] C. Derichs and H. Niemann, "Handling camera movement constraints in reinforcement learning based active object recognition," in *Proc. Joint Pattern Recognit. Symp.*, 2006, pp. 637–646.

[29] H. Liu, H. Liu, F. Sun, and B. Fang, "Kernel regularized nonlinear dictionary learning for sparse coding," *IEEE Trans. Syst., Man, Cybern., Syst.*, to be published, doi: 10.1109/TSMC.2017.2736248.

[30] J. Peters, K. Mülling, and Y. Altün, "Relative entropy policy search," in *Proc. AAAI*. Atlanta, Georgia, 2010, pp. 1607–1612.

[31] T. P. Lillicrap *et al.*. (2015). "Continuous control with deep reinforcement learning." [Online]. Available: https://arxiv.org/abs/1509.02971