Belief Tree Search for Active Object Recognition

Mohsen Malmir[1] and Garrison W. Cottrell[2]

Abstract— **Active Object Recognition (AOR) has been approached as an unsupervised learning problem, in which optimal trajectories for object inspection are not known and to be discovered by reducing label uncertainty or training with reinforcement learning. Such approaches suffer from local optima and have no guarantees of the quality of their solution. In this paper, we treat AOR as a Partially Observable Markov Decision Process (POMDP) and find near-optimal values and corresponding action-values of training data using Belief Tree Search (BTS) on the AOR belief Markov Decision Process (MDP). AOR then reduces to the problem of knowledge transfer from these action-values to the test set. We train a Long Short Term Memory (LSTM) network on these values to predict the best next action on the training set rollouts and experimentally show that our method generalizes well to explore novel objects and novel views of familiar objects with high accuracy. We compare this supervised scheme against guided policy search, and show that the LSTM network reaches higher recognition accuracy compared to the guided policy search and guided Neurally Fitted Q-iteration. We further look into optimizing the observation function to increase the total collected reward during active recognition. In AOR, the observation function is known only approximately. We derive a gradient-based update for the observation function to increase the total expected reward. We show that by optimizing the observation function and retraining the supervised LSTM network, the AOR performance on the test set improves significantly.**

I. INTRODUCTION

Active Object Recognition (AOR) refers to the problem of predicting object label from the images while being able to change the pose of the object relative to the camera for increasing prediction certainty. A robot rotating an in-hand object to refine its label prediction accuracy is an example of an AOR system. Ambiguity in object recognition exists because of similar views of different objects. AOR aims at finding the optimal sequence of actions which decreases the label ambiguity and improves object recognition performance in smaller number of steps. Despite its wide application and performance improvement capacity, AOR has not been applied widely and has remained secluded from main-stream computer vision progress in recent years.

Existing approaches to AOR change the sensor position to reduce the ambiguity of label prediction [1], [4], [9]. Most of these methods rely on uncertainty about object label, and use greedy best next action selection [7] at the test time to decrease label probability entropy. A few methods aim for optimal action selection at the test time using dynamic programming [2] or Monte Carlo planning [24]. However these methods require a model of the object, and are computationally heavy at the test time. We propose a method that reduces optimal action selection to classification of belief at the test time and does not require a generative model of objects. We show that the proposed method generalizes well to *novel views* of familiar objects, and also to *novel* objects. Moreover, we show that AOR paradigm can be used to improve the label prediction accuracy of image classifier by emphasizing images in the train set that are more likely to result in higher rewards.

In the first contribution of this paper, we formulate AOR as a POMDP problem and adapt a Belief Tree Search algorithm [16] to discover near-optimal values for objects poses on the training set. We infer a policy from these values, and use it to train an LSTM network to predict the best action given the current objects belief. At the test time, we use the actions predicted by this LSTM to explore the objects. We show that this supervised approach generalizes well to explore novel objects and novel views of familiar objects, and results in higher AOR accuracy compared to reinforcement learning and guided policy search methods.

In our second contribution, we derive an update rule to learn the parameters of the POMDP likelihood function with the goal of maximizing the total reward. This update rule emphasizes views of objects that will produce higher rewards in the future. We show that by retraining the likelihood function using the proposed method, the performance of the AOR system significantly improves.

In the next section, we review the previous approaches related to our method. Then we present the BTS algorithm and the observation function update rule. In the results section, we report the details of the implementation of these methods and their performance on GERMS [18] dataset. GERMS has shown to be a challenging AOR dataset, and we improve state-of-the-art performance on this dataset. The final section is the concluding remarks.

II. LITERATURE REVIEW

A large category of AOR models try to minimize the predicted label uncertainty through best next-action planning [28], [5], [8], [9], [6], [7]. These models predict the object label probabilities using the current view, and search for the best next action that minimizes the expected entropy of object labels. In these methods, learning object appearance is performed by fitting a generative model offline, while best action selection is carried out online at the test time. Uncertainty measures such as conditional entropy and mutual information are computationally expensive to evaluate for all

[1]Mohsen Malmir is a PhD candidate at the Computer Science and Engineering Department, University of California San Diego, 9500 Gilman Dr., San Diego, CA, 92093 mmalmir@eng.ucsd.edu

[2]Garrison W. Cottrell is a professor at the Computer Science and Engineering Department, University of California San Diego, 9500 Gilman Dr., San Diego, CA, 92093 gary@eng.ucsd.edu

possible observations. Therefore these methods usually resort to approximations of these measures, which might result in poor AOR performance.

A second category of models use techniques such as REINFORCE [31] or Neurally Fitted Q-Iteration (NFQ) [27] to find a good policy or action-value function for object exploration [23], [20], [18]. A parametric function that encodes object exploration policy or action-values is learned offline by using an exploration policy and collecting rewards that depend on the label prediction accuracy. The model then updates the parameters of policy or action-value function to maximize its total expected reward. These method suffer from high variance of prediction due to sampling of actions, only guarantee convergence to a local optima and require large training periods to explore and discover suitable sequences of actions.

More recently, deep convolutional neural networks (CNN) have been applied in AOR as a tool for modeling object appearance along with action-value prediction [14], [17], [10], [12]. Malmir et. al trained a deep CNN using NFQ update rule [17]. In this work, a layer of Dirichlet distribution is embedded into the network for modeling the distribution of beliefs for different object-action pairs. Johns et. al used deep CNNs for entropy regression and action prediction for the set of next view points [14]. Finding the optimal trajectory for object inspection is then approximated by maximizing the sum of cross entropy over adjacent views pairs. Haque et al. trained LSTM networks with REINFORCE algorithm to recognize subjects from 3D point-clouds [10]. Jayaraman and Grauman modeled object exploration policy as a neural network and trained it using classification accuracy as reward [12]. They found that predicting the next state of the environment based on current state and action improves the overall AOR accuracy. All these methods show improved performance over random exploration strategy and non-active methods. However they suffer from the same problem as previous methods, which is lack of guarantee of performance even on the training set.

There are very few approaches to AOR that aim to find optimal exploration policies. Atanasov et al. [2] adapted an active hypothesis testing approach [22] for camera view-point selection for object segmentation. This approach learns a model of object appearance, and uses that for planning a sequence of actions that minimizes the cost of motor movements, object classification and view-point prediction. A dynamic programming approach is used to discover the best sequence of actions that minimizes this cost. This method depends on the representation of object appearance for efficient planning, while our method acts in the belief space and is completely independent of object representation and classification. Patten et al. used Monte Carlo planning for active exploration and perception of outdoor objects [24]. This method uses rollouts that depends on the point-cloud of objects for different actions. Compared to this method, our method is more intuitive, and doesn't require 3D models of objects.

Of special interest to this paper are works on belief tree search and Monte Carlo POMDP planning. Lee et. al [16] proposed clustering of beliefs in a belief tree search algorithm to reduce the width of the tree. DESPOT [30] uses sampling of observations to reduce the width of the belief tree for optimal action selection. POMCP [29] adapts Monte Carlo sampling and the Upper Confidence Trees algorithm [15] for efficient POMDP planning. We adapt an approach similar to [16] because of desirable properties such as acting on the belief space and performance guarantees on the *reachable* space of beliefs.

An important property of the proposed approach is its freedom of maintaining an object model. Our method acts in the belief space of objects, and only requires a black box simulator of objects that produces rollouts of object examination, and returns the sequence of actions and their corresponding beliefs. Another important property of our method is that at the test time, the next action selection is achieved by (computationally simple) classification of the current belief. We show that this approach is more effective for learning object exploration policies, compared to reinforcement learning and actor-critic methods. In the next section we describe the proposed approach in more details.

III. Proposed Method

A. Belief Tree Search

Active object recognition is formulated as a POMDP problem denoted with tuple $< \mathcal{S}, \mathcal{A}, \mathcal{O}, \mathcal{T}, \mathcal{P}, \mathcal{R}, \gamma >$, where $\mathcal{S}$ is the set of states (object labels), $\mathcal{A}$ is the set of actions for object examination, and $\mathcal{O}$ is the set of observations (captured images of objects). We don't perceive the identity of object directly but rather collect information about it through observations. The transition function $\mathcal{T}$ marks the transition between different states and the observation function $\mathcal{P}(s, a, o)$ relates the observations to different object identities through $\mathcal{P}(s, a, o) = Pr(o|s, a)$, the probability of observing o after taking action a when the object labels is s. The reward function $\mathcal{R}(s, a)$ determines the reward for taking action a when the object label is s. Finally, γ is the reward discount factor.

In AOR POMDP, the transition function $\mathcal{T}$ reduces to the identity function. The observation space on the other hand includes all images of objects, and is prohibitively large to apply value iteration techniques [26]. Another possibility is to use Monte Carlo planning , which builds a search tree for one-step action planning [29]. However this method suffers from the curse of history. We seek to find a solution to the AOR POMDP that compactly represents history, and allows tractable search for finding the optimal values, and the corresponding action-values.

It has been shown that solution to POMDP can be found by solving the equivalent *belief* MDP, where the belief $b \in \mathbb{R}^{|\mathcal{S}|}$ denotes the posterior probability of states given the observation history,

$$b^t(s) = Pr(s|a_{0:t-1}, o_{0:t-1}) \tag{1}$$
$$\sum_{s \in \mathcal{S}} b(s) = 1, \;\; b(s) \geq 0 \;\; \forall s \in \mathcal{S}.$$

In belief MDP, states represent the posterior probabilities of POMDP states given the action-observation history. For this MDP, the transition between beliefs given action a is defined as,

$$\begin{aligned}&\mathcal{T}^{MDP}(b,a,b')\\&= Pr(b'|b,a)\\&= \sum_{o\in\Omega(b,a,b')}\sum_{s,s'\in\mathcal{S}} b(s)\mathcal{T}^{POMDP}(s,a,s')\mathcal{P}(s',a,o)\end{aligned} \quad (2)$$

where $\Omega(b,a,b')$ denotes the set of observations that can result in changing beliefs from b to b',

$$\begin{aligned}&\Omega(b,a,b') = \{o\in\mathcal{O}|\\&\quad b'(s') = \frac{\mathcal{P}(s',a,o)\sum_{s\in\mathcal{S}}\mathcal{T}^{POMDP}(s,a,s')b(s)}{Pr(o|a,b)}\}\end{aligned} \quad (3)$$

The belief MDP reward is defined by calculating the expected reward over states,

$$R(b,a) = \sum_{s\in\mathcal{S}} b(s)\mathcal{R}_s^a. \quad (4)$$

And finally, given an initial belief b, action a and observation o the updated belief b' is calculated by,

$$\begin{aligned}b'(s') &= Pr(s'|b,a,o)\\&= \frac{\mathcal{P}(a,s',o)\sum_{s\in\mathcal{S}} b(s)\mathcal{T}^{POMDP}(s,a,s')}{Pr(o|a,b)}.\end{aligned} \quad (5)$$

Now that we defined the equivalent belief MDP, we aim at solving the planning problem using *Belief Tree Search* algorithm. This algorithm constructs a search tree for a given belief b, where different branches represent actions and the resulting observations. Each node in the tree represents a belief about the underlying POMDP states and edge captures an action and the resulting observation. The algorithms starts with b at the root and exhaustively performs all action to collect new observations and form new beliefs. These beliefs are then added as children of the root and the process iterates with new beliefs until stop states are reached in the leaves. The values are then backtracked from the leaves to the root to estimate the value of b. The belief tree search is used in online planning for POMDPs where the dynamics of the environment are known. We adapt BS to calculate the optimal values of images for the training set.

The plain belief tree search algorithm is computationally intractable to use for AOR belief MDP, since it examines all observations for each action. Instead we adapt the algorithm in theorem 1 of [16], which sacrifices optimality of the predicted values in exchange of computational tractability. This algorithm utilizes the smoothness of the optimal value function to cluster the belief space. More specifically, for a given $\delta > 0$, each node in the tree represents the approximate value of beliefs that are in δ-neighborhood of b', which we represent as $\delta(b)$. By clustering the belief space, the width of the belief tree decreases to a manageable size. This algorithm calculates the approximate optimal value of a given belief b only in $\mathcal{R}(b)$, which is the *reachable space* of b. The reachable space is defined as the set of beliefs that are reachable by arbitrary sequences of actions from b. Finally, the algorithm utilizes the discount factor γ to limit the height of the belief tree.

We adapt this algorithm to find an approximately optimal value for each image in the training set. The algorithm is depicted in pseudo code style in algorithm 1. Using these values, training an active object recognition system reduces to a supervised learning and knowledge transfer problem. In each node of the tree, the algorithm expands all actions and receives the new observations. New beliefs are then calculated using (5). For each new belief b', if there is an already expanded belief b_0 in that height of the tree for which $b' \in \delta(b_0)$, the algorithm sets the value of b' equal to b_0 and backtracks. Otherwise, the search continues in the children of b'.

Our algorithm builds a belief tree over $\mathbf{R}(b_0)$ by sampling from images in the training set and maintaining a δ-packing of $\mathbf{R}(b_0)$ at each level of the tree. A belief tree with root b_0 denotes all the possible actions and observations that are encountered while inspecting an object with the initial belief b_0. A belief tree captures all the possible actions and observations, construction of full belief tree is prohibitive in case of active object recognition because the size of the observation space is extremely large. One modification that we made to algorithm 1 compared to theorem 1 of [16] is that at the root node, we calculate the value of $\delta(b_0)$. This is to reduce the overfitting of values to specific beliefs. In our algorithm, if two beliefs are very similar but result in vastly different rewards, that should be considered in calculating the value of b_0. In AOR this happens when the classifier is uncertain about some examples then their beliefs are close to each other and this should be reflected in their calculated values. Figure 1 demonstrates the belief tree search .

Algorithm 1 Belief Tree Search

Belief Tree Search(Belief b_0, radius δ, max height h)
for All Images o_i in training set **do**
  $b_i \leftarrow$ Classify(o_i)
  if $b_i \in \delta(b_0)$ **then**
    Expand($o_i, b_i, 0$)
  end if
end for

Expand(Image o, Belief b, level i)
if $i = h$ **then return**
end if
for action $a \in \mathcal{A}$ **do**
  Image $o_a \leftarrow$ Simulate-Action(o, a)
  Belief $b_a \leftarrow$ Transition(b, a, o_a)
  if $b_a \in \delta$-neighborhood of any b' belief at level $i+1$ **then**
    $V(b_a) \leftarrow V(b')$
  else
    Add b_a to nodes in level $i+1$
    Expand($o_a, b_a, i+1$)
  end if
  add (o_a, b_a) to children of b
end forreturn

Theorem 1 provides guarantee on the optimality of the values found for images in algorithm (1).

Theorem 1. For a given maximum error ϵ, and the optimal value function $V^*(b)$, algorithm (1) finds the value of a given belief b_0 such that $|V^*(b_0) - V(b_0)| \leq \epsilon$ by setting the

parameter values as,

$$\delta = \frac{\epsilon(1-\gamma)^2}{2R_{max}} \tag{6}$$

$$h = \log_\gamma \frac{(1-\gamma)\epsilon}{2R_{max}} \tag{7}$$

Proof. See the supplementary materials[1].

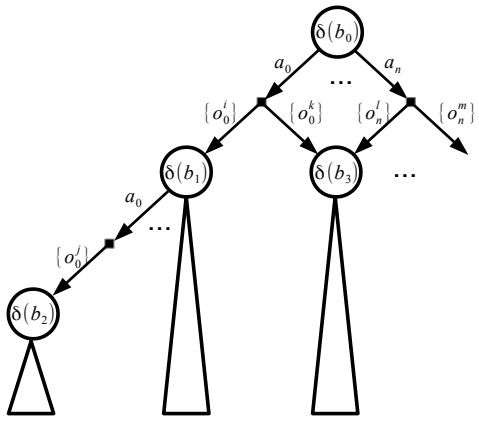

Fig. 1. Belief Tree Search algorithm. Each node in the tree represents the value of δ-neighborhood of a belief with some error. Different observations may lead to visiting the same belief after taking an action.

B. Optimizing Observation Function

In POMDP problems, usually it is assumed that the observation function is given as part of the environment. In AOR POMDP, the observation function is usually modeled using a generative model of objects. For example Borotsching et. al use Gaussian mixture on the eigenspace to model the likelihood of images from the view sphere of object under different classes [5]. Calculating the observation function value for an image usually requires feature extraction from the image and density estimation for different classes. We assume an observation function parameterized by ϕ,

$$\mathcal{P}(s,a,o;\phi) = Pr(o|s,a;\phi) \tag{8}$$

where ϕ is usually learned using maximum likelihood estimation. Different values of parameters ϕ changes the observation function. One may *improve* the observation function by using a different estimation of ϕ, which changes the feature extraction or density estimation. The improved observation function then results in a POMDP with different environment dynamics. In the ideal case, the observation function for an image is 1 for the correct object label and 0 for other labels, in which case the POMDP reduces to a trivial MDP.

For a given observation function, theorem 1 finds the image values for policies arbitrarily close to the optimal policy. However these values depend on the POMDP dynamics, e.g. the observation function. We propose to improve the observation function by increasing the total reward collected by the near-optimal policy. For any policy π and observation function $\mathcal{P}$, the total reward ρ is defined as,

$$\begin{aligned}\rho(\pi,\mathcal{P}) &= E\left\{\sum_{t=1}^{\infty}\gamma^t r_t|s_o,\pi\right\} \\ &= \int_{b\in\mathcal{B}} d^{\mathcal{P},\pi}(b)\sum_{a\in\mathcal{A}}\pi(b,a)R(b,a)db\end{aligned} \tag{9}$$

Where $\mathcal{B}$ is the $|\mathcal{S}|$-dimensional belief simplex. Changing the observation function parameters may increase the likelihood of images under the correct object label. Theorem 2 presents a gradient ascent update rule to the parameters of observation function, with the goal of increasing the total reward.

Theorem 2. Given a policy π and the corresponding value function V^π the gradient of the total reward $\rho(\pi,\mathcal{P})$ with respect to the parameters of the observation function $\mathcal{P}(;\phi)$ is given by,

$$\begin{aligned}&\frac{\partial}{\partial\phi}\rho(\mathcal{P},\pi) \\ &= \int_b d^{\mathcal{P},\pi}(b)\sum_a\pi(b,a)\int_{b'}\sum_{o\in\Omega(b,a,b')}\sum_{s,s'\in\mathcal{S}} \\ &\quad b(s)T(s,a,s')V^\pi(b')\frac{\partial}{\partial\phi}\mathcal{P}(o|s',a)\mathrm{d}b'\mathrm{d}b\end{aligned} \tag{10}$$

Proof. See the supplementary notes.

Intuitively speaking, the update rule in (10) weights each parameter by the value of the belief that is reached by observing the corresponding o. Changing the observation function parameters ϕ changes the belief MDP dynamics as the transition probabilities in (2) depend on $\mathcal{P}$. After updating the observation function, a new belief MDP is reached, for which we can use theorem 1 to find near-optimal values of images. In practice evaluation of (10) is computationally intractable. In the results section, we use a sampling approach for updating the observation function based on the values of images.

IV. Results

A. Calculating BTS Action-Values

In this section, we implement the proposed method in algorithm (1) for active recognition of GERMS [18]. This is a medium size dataset with $\sim 120K$ images of 136 different object collected by a robot. The robot grabs each object with 10 different orientations and examines the object by rotating them in front of the camera. The goal is to recognize object for 4 test orientations, given the other 6 in-hand orientations. The actions bring the robot gripper into one of 5 pre-defined positions that are evenly spaced in the rotation range (180 degrees) of the gripper. GERMS is proved to be a challenging dataset since separation of objects in this dataset requires extraction of fine-grained visual cues.

We extract visual features from GERMS images using ResNet deep CNN model [11]. A softmax layer is trained on top of these features to predict the object label. Then we convert train and test images into belief vectors, and train our AOR method in the belief space. We normalize the output of

[1]https://sites.google.com/a/eng.ucsd.edu/mmalmir/BTSSupp.pdf

the softmax layer for each class to sum to 1 over all GERMS images and use that as the observation probability. This is to ensure that the deep CNN outputs the likelihood and to maintain the integrity of (2) and (5).

After calculating the likelihood of train images, we use algorithm 1 to obtain the value of the near-optimal policy for them. In order to use these values in planning, [16] proposes a sampling approach that repeatedly executes algorithm 1 for different simulations and augments the tree with newly discovered beliefs and finally uses the action-values of the root of the resulting tree for planning. The proposed BTS algorithm is similar to the work in [16] in that we use belief vectors in δ-neighborhood of the root to run the simulations. We found the action-values of the root of the tree to be very effective for AOR.

After we extract the action-values for training images, we transfer the knowledge of these action-values to the test set using three different approaches. In the first and second approaches, we use *Neurally Fitted Q-learning* (NFQ) [27] and *Actor Critic* (AC) [25], guided by a probabilistic policy that uses the action-values from BTS. We show that guiding NFQ and AC results in slight improvement of average performance on the test set, compared to the plain version. In the third approach, we use an LSTM network to learn to predict the best action given the objects belief.

B. Guided Neurally Fitted Q-learning

Neurally Fitted Q-learning (NFQ) trains a neural network to predict the action-values using the reward signal from the environment [27]. This algorithm has been successfully applied to reinforcement learning benchmarks [27], Atari games [21] and active object recognition [18]. At the heart of this approach is an iterative update rule for the network parameters θ,

$$\theta_{t+1} \leftarrow \theta_t + \alpha \frac{\partial}{\partial \theta} (R_t + \gamma \max_a Q(s_{t+1}, a) - Q(s_t, a_t))^2 \tag{11}$$

where R_t is the reward at time step t, γ is the reward discount factor, α is the learning rate and $Q(s, a)$ is the network output denoting action-values for action a in state s. In the above, the gradient operator on the right hand applies only to $Q(s_t, a_t)$. It has been observed that the plain NFQ algorithm may fail to discover optimal policies for active object recognition [17]. Instead, we employ the n-step extension of this algorithm, proposed in [19], in which the update rule in (11) is applied to action sequences of length n. The n-step NFQ speeds up learning by updating n action-values in each iteration, compared to a single action-value update in the original NFQ. All experiments reported here are obtained using 10-step rollouts.

We improve the performance of NFQ by applying the *importance sampling* framework for policy search [13]. The idea is to use an auxiliary policy $\pi_{\theta'}$ to acquire sequences of actions and states $\tau = \{s_0, a_0, s_1, a_1, \ldots\}$, and update the parameters of the target policy π_θ using these sequences. In order to obtain an unbiased estimate of the policy gradients, update terms are weighted by their importance,

$$\frac{P(\tau|\pi_\theta)}{P(\tau|\pi_{\theta'})} = \frac{\prod_j \pi_\theta(a_j|s_j)}{\prod_j \pi_{\theta'}(a_j|s_j)} \tag{12}$$

Where $P(\tau|\pi)$ is the probability of rollout sequence τ under policy π, and $\pi(a|s)$ is the probability of action a in state s under policy π. We implement the guided NFQ (GNFQ) by drawing sequences of actions from an stochastic policy acquired by performing softmax on BTS action-values. The gradients in (11) are then multiplied by their importance in (12) and applied to the network. Figure 2 shows the performance of NFQ and GNFQ on the GERMS test set. For both approaches, we show $1-\sigma$ interval of object recognition accuracy. We also report the performance of random policy (RND), where at each time step, a previously *unseen* view of the object is explored. We see that the plain NFQ algorithm fails to surpass the random policy, while GNFQ performs slightly better than random. The advantage of GNFQ over NFQ and RND is most significant over the first action, and it gradually fades over the next four actions. After the last action where the majority of evidence has been accumulated, all three methods reach the same label prediction accuracy.

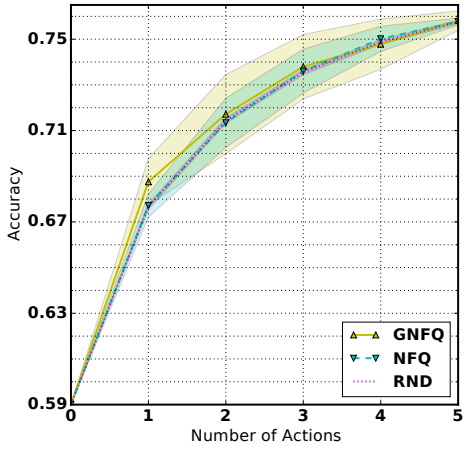

Fig. 2. Comparison of AOR performance for NFQ, guided NFQ and random policies. The shaded areas show accuracy mean $\pm$ std.

C. Guided Actor-critic

Actor-critic is a policy learning method which updates the policy parameters using gradients of the total expected reward [25]. To reduce the variance of gradient estimation, a bias which is the prediction of the current state value, is decreased from the reward,

$$\theta_t \leftarrow \theta_t + \alpha \frac{\partial}{\partial \theta} \log \pi_\theta(a_t|s_t)(R_t - V_\psi(s_t)) \tag{13}$$

$$\psi_t \leftarrow \psi_t + \alpha \frac{\partial}{\partial \psi} (V_\psi(s_t) - R_t)^2 \tag{14}$$

where α is the learning rate, R_t is the reward at time t, $V_\psi(s_t)$ is the predicted value for s_t parameterized by ψ, and

$\pi_\theta(a_t|s_t)$ is the probability that policy π_θ, parameterized by θ, assigns to action a_t in state s_t. In figure 3, we show the performance of 10-step actor-critic method with (ACG) and without (AC) guiding. We used the same guiding scheme as described above, by multiplying the gradient terms of (13) and (14) by their importance (12). We see that the plain AC fails to perform better than random, while ACG shows slight performance improvements over RND after the second and third actions.

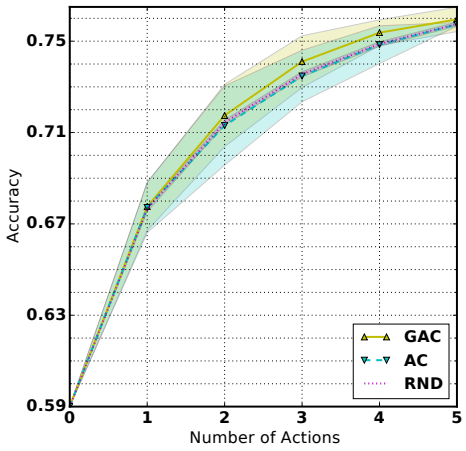

Fig. 3. Comparison of AOR performance for actor-critic, guided actor-critic and random policies. The shaded areas show accuracy mean $\pm$ std.

D. Supervised Learning of Action-Values

To transfer the knowledge of extracted action-values from training to test set, we train a neural network to directly predict the best next action given a belief. We use a stack of 3 LSTM layers with 128 units in each layer, followed by a softmax layer that predicts action (see Figure 4). The training sequences are produced by following a probabilistic policy derived from BTS action-values. For each belief vector, the target action is the action selected in the rollout by the BTS-driven policy. We found that using both rollouts and LSTM is crucial to the AOR performance in supervised action prediction. Figure 5 compares the performance of LSTM and random policies. LSTM has a clear advantage in performance over the random policy. Moreover, the variance of the learned policy is significantly smaller compared to actor-critic and NFQ methods. Figure 6 compares the average performance of all methods. We see that supervised learning for action-value prediction is clearly superior to the policy learning methods. Table I compares the performance of all methods in more details.

E. Generalization to Novel Objects

In this section, we test the generalization of the proposed AOR method to novel object. The goal of this experiment is to understand how much object inspection knowledge is transferrable between GERMS objects. For this purpose, we use 60% of objects in GERMS for supervised training of action prediction, and the rest of objects for testing. The results averaged over 20 different experiments are shown in figure 7. Overall, the variance of results is high because of the large variations in the recognition accuracy of GERMS objects. We see that the supervised LSTM method achieves significantly higher performance compared to the random strategy.

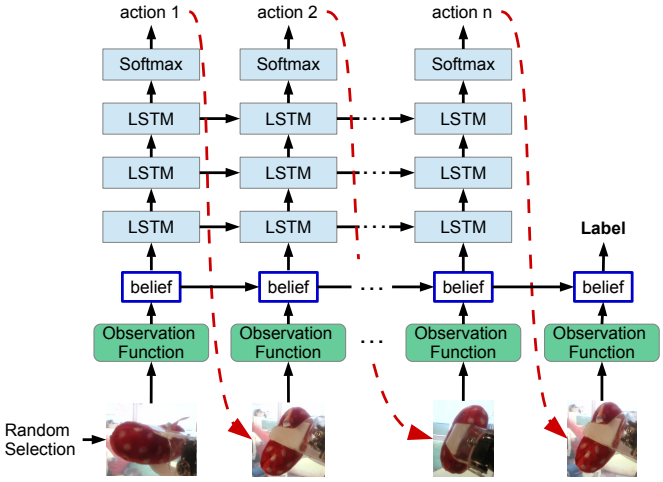

Fig. 4. The proposed supervised learning method for action prediction using LSTM network. The observation function is fixed for training the LSTM layers.

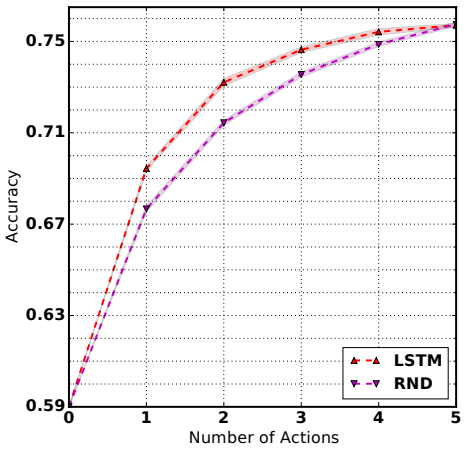

Fig. 5. Comparison of AOR performance for LSTM and random methods. The shaded areas show accuracy mean $\pm$ std.

F. Improving Observation Function

We retrain the observation function using the proposed gradient update rule in (10) to increase the AOR total collected reward. In order to implement this update rule, we adapt a sampling strategy by generating a set of rollouts using policy π derived from BTS action-values. Let b_i denote the belief vector corresponding to image I_i. Let $\{b_j, a_j\}, j = 1, 2, \ldots, n$ denote the set of beliefs and actions that resulted in b_i in the rollouts. The retraining weight of I_i is then calculated using a sample average of (10) on these rollouts,

$$Weight(b_i) \propto \frac{1}{n} \sum_{j=1}^{n} \pi(b_j, a_j) b_j(s) V^\pi(b_i) \qquad (15)$$

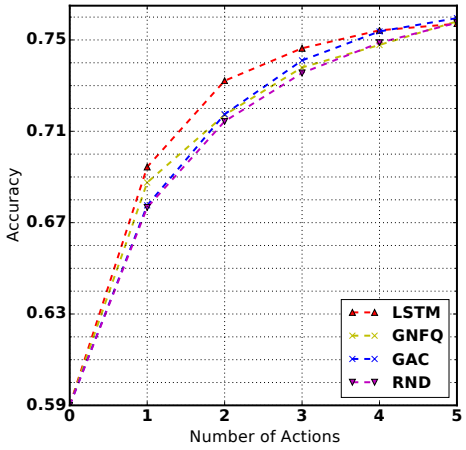

Fig. 6. Comparison of AOR performance for LSTM, AC, NFQ and RND policies.

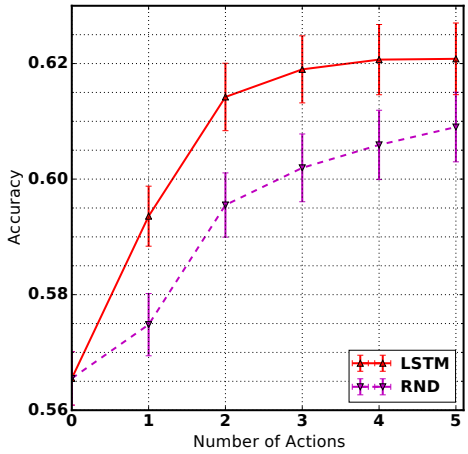

Fig. 7. Generalization of object inspection policies of LSTM model to novel objects. The reported numbers are mean± standard error.

TABLE I

AOR PERFORMANCE COMPARISON ON THE GERMS TEST DATA BASED ON THE NUMBER OF ACTIONS.

method	0	1	2	3	4	5
RND	0.590	0.677	0.714	0.736	0.749	0.758
AC	0.590	0.677	0.713	0.735	0.748	0.757
ACG	0.590	0.678	0.717	0.741	0.754	0.760
NFQ	0.590	0.677	0.713	0.736	0.750	0.758
NFQG	0.590	0.688	0.717	0.738	0.748	0.758
LSTM	0.590	0.694	0.732	0.746	0.754	0.757
LSTM-i2	0.614	0.715	0.751	0.769	0.778	0.785
LSTM-i3	**0.617**	**0.718**	**0.758**	**0.776**	**0.790**	**0.793**

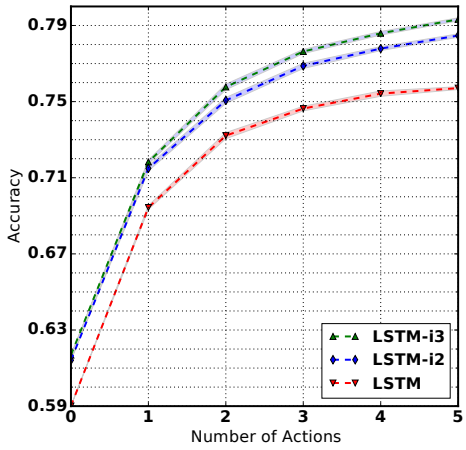

Fig. 8. comparison of the accuracy of the retrained LSTM models, denoted as LSTM-iteration2 (LSTM-i2) and LSTM-iteration3 (LSTM-i3) with the original model.

where V^{π} values are calculated using algorithm 1, $b(s)$ denotes the entry corresponding to object label s in belief vector b, and $\pi(b, a)$ is the probability assigned to action a for belief b by the BTS-driven policy. We retrain the softmax over visual features, by weighting the cross entropy cost of each image by (15). After retraining, we run the BTS algorithm on the resulting beliefs and train the LSTM model on the resulting action-values. We show the performance of the retrained LSTM as *LSTM-i2* in figure 8. The retrained LSTM achieves significantly higher accuracy compared to the original LSTM. We repeat this procedure and observe slight improvements, shown as *LSTM-i3* in figure 8. In practice, the performance of the LSTM starts to decline after the second retraining. The performance of the retrained models are shown in more details in table I.

V. CONCLUSIONS

Active object recognition has received little attention from mainstream computer vision and machine learning communities despite its potential for improving recognition performance. Progress on AOR has been slow due to reliance of the AOR models on semi supervised or heuristic methods for learning object inspection policy. In this work, we proposed a method that learns the object exploration policy in a supervised manner from the training data. The proposed method has desirable properties, for example it provides guarantee of optimality of calculated values on training set, it is very fast at the test time and generalizes well to novel objects and novel views of familiar objects since it does not require a generative model of objects.

Recently there has been a renewed interest in attention models mostly for reasons beyond object recognition. These models rely on reinforcement learning [20] or variation inference [3] to guide the system to discover suitable exploration policies. In contrast, our method benefits from reduced training time, which is a large problem in these approaches. By reducing the optimal policy inference to a supervised learning problem, we can use recent advances in supervised visual recognition for learning policies and knowledge transfer to test data.

We developed a weighting scheme for training classification of single images, that puts emphasize on images with higher values during exploration. The weight of each image is a measure of how useful it is in inferring the correct label of object. Such weighting scheme potentially reduces overfitting of the image classification model to the

training data that have little or no discriminative information. Because of the complex background in GERMS images and blocking of in-hand object by the robot gripper, there is high chance of overfitting to background cues during training. By performing BTS, the AOR has the opportunity to look ahead a few steps in active inspection of objects. Image values are then used to guide the single image classification model to discover more discriminative features.

REFERENCES

[1] J. Aloimonos, I. Weiss, and A. Bandyopadhyay. Active vision. *International journal of computer vision*, 1(4):333–356, 1988.

[2] N. Atanasov, B. Sankaran, J. Le Ny, G. J. Pappas, and K. Daniilidis. Nonmyopic view planning for active object classification and pose estimation. *IEEE Transactions on Robotics*, 30(5):1078–1090, 2014.

[3] J. Ba, V. Mnih, and K. Kavukcuoglu. Multiple object recognition with visual attention. *arXiv preprint arXiv:1412.7755*, 2014.

[4] R. Bajcsy. Active perception. *Proceedings of the IEEE*, 76(8):966–1005, 1988.

[5] H. Borotschnig, L. Paletta, M. Prantl, and A. Pinz. Appearance-based active object recognition. *Image and Vision Computing*, 18(9):715–727, 2000.

[6] B. Browatzki, V. Tikhanoff, G. Metta, H. H. Bülthoff, and C. Wallraven. Active object recognition on a humanoid robot. In *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, pages 2021–2028. IEEE, 2012.

[7] B. Browatzki, V. Tikhanoff, G. Metta, H. H. Bulthoff, and C. Wallraven. Active in-hand object recognition on a humanoid robot. *Robotics, IEEE Transactions on*, 30(5):1260–1269, 2014.

[8] F. G. Callari and F. P. Ferrie. Active object recognition: Looking for differences. *International Journal of Computer Vision*, 43(3):189–204, 2001.

[9] J. Denzler, C. M. Brown, and H. Niemann. Optimal camera parameter selection for state estimation with applications in object recognition. In *Pattern Recognition*, pages 305–312. Springer, 2001.

[10] A. Haque, A. Alahi, and L. Fei-Fei. Recurrent attention models for depth-based person identification. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[11] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[12] D. Jayaraman and K. Grauman. Look-ahead before you leap: end-to-end active recognition by forecasting the effect of motion. In *European Conference on Computer Vision*, pages 489–505. Springer, 2016.

[13] T. Jie and P. Abbeel. On a connection between importance sampling and the likelihood ratio policy gradient. In *Advances in Neural Information Processing Systems*, pages 1000–1008, 2010.

[14] E. Johns, S. Leutenegger, and A. J. Davison. Pairwise decomposition of image sequences for active multi-view recognition. *arXiv preprint arXiv:1605.08359*, 2016.

[15] L. Kocsis and C. Szepesvári. Bandit based monte-carlo planning. In *European conference on machine learning*, pages 282–293. Springer, 2006.

[16] W. S. Lee, N. Rong, and D. J. Hsu. What makes some pomdp problems easy to approximate? In *Advances in neural information processing systems*, pages 689–696, 2007.

[17] M. Malmir, K. Sikka, D. Forster, I. Fasel, J. R. Movellan, and G. W. Cottrell. Deep active object recognition by joint label and action prediction. *Computer Vision and Image Understanding*, pages –, 2016.

[18] M. Malmir, K. Sikka, D. Forster, J. Movellan, and G. Cottrell. Deep q-learning for active recognition of germs: Baseline performance on a standardized dataset for active learning. In *Proceedings of the British Machine Vision Conference (BMVC)*, pages 161.1–161.11. BMVA Press, September 2015.

[19] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *International Conference on Machine Learning*, pages 1928–1937, 2016.

[20] V. Mnih, N. Heess, A. Graves, and K. Kavukcuoglu. Recurrent models of visual attention. In *Advances in Neural Information Processing Systems*, pages 2204–2212, 2014.

[21] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015.

[22] M. Naghshvar, T. Javidi, et al. Active sequential hypothesis testing. *The Annals of Statistics*, 41(6):2703–2738, 2013.

[23] L. Paletta and A. Pinz. Active object recognition by view integration and reinforcement learning. *Robotics and Autonomous Systems*, 31(1):71–86, 2000.

[24] T. Patten, W. Martens, and R. Fitch. Monte carlo planning for active object classification. *Autonomous Robots*, pages 1–31, 2017.

[25] J. Peters and S. Schaal. Reinforcement learning of motor skills with policy gradients. *Neural networks*, 21(4):682–697, 2008.

[26] J. Pineau, G. Gordon, S. Thrun, et al. Point-based value iteration: An anytime algorithm for pomdps. In *IJCAI*, volume 3, pages 1025–1032, 2003.

[27] M. Riedmiller. Neural fitted q iteration–first experiences with a data efficient neural reinforcement learning method. In *European Conference on Machine Learning*, pages 317–328. Springer, 2005.

[28] B. Schiele and J. L. Crowley. Transinformation for active object recognition. In *Computer Vision, 1998. Sixth International Conference on*, pages 249–254. IEEE, 1998.

[29] D. Silver and J. Veness. Monte-carlo planning in large pomdps. In *Advances in neural information processing systems*, pages 2164–2172, 2010.

[30] A. Somani, N. Ye, D. Hsu, and W. S. Lee. Despot: Online pomdp planning with regularization. In *Advances in neural information processing systems*, pages 1772–1780, 2013.

[31] R. J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3-4):229–256, 1992.