Reinforcement Learning of Active Vision for Manipulating Objects under Occlusions

Ricson Cheng, Arpit Agarwal, Katerina Fragkiadaki
Carnegie Mellon University
United States
{ricsonc,arpita1}@andrew.cmu.edu, katef@cs.cmu.edu

Abstract: We consider artificial agents that learn to jointly control their gripper and camera in order to reinforcement learn manipulation policies in the presence of occlusions from distractor objects. Distractors often occlude the object of interest and cause it to disappear from the field of view. We propose hand/eye controllers that learn to move the camera to keep the object within the field of view and visible, in coordination to manipulating it to achieve the desired goal, e.g., pushing it to a target location. We incorporate structural biases of object-centric attention within our actor-critic architectures, which our experiments suggest to be a key for good performance. Our results further highlight the importance of curriculum with regards to environment difficulty. The resulting active vision / manipulation policies outperform static camera setups for a variety of cluttered environments.

Keywords: Manipulation, Reinforcement learning, Control

1 Introduction

We consider artificial agents that learn to perform manipulation tasks in cluttered environments. In this case, state estimation, namely the prediction of 3D object locations and poses, is particularly challenging due to frequent occlusions of the target object from distractors. We, humans, move our head to find the best viewpoint for the actions we wish to carry out. We turn our heads towards the direction we expect the object to move, and actively select viewpoints to facilitate perception during manipulation. Yet, reinforcement learning has thus far almost exclusively considered static cameras. If the agent was given the opportunity to (learn to) move its camera, potentially better manipulation policies would be found, exploiting the easier state estimation thanks to dis-occlusions of the object of interest by active vision policies [1].

We explore actor-critic architectures for hand/eye control in cluttered scenes. We specifically consider pushing an object to target locations in environments with distractors; the distractors often cause the object of interest to be occluded. The proposed architectures learn camera control policies that facilitate state estimation by learning to increase visibility of the main object the agent interacts with, in coordination to the manipulation actions of the agent. Our experiments suggest two important and surprising facts: First, training with vanilla CNN architectures failed to learn the task of object pushing in the presence of even simple distractors. Incorporating a trained object detector module in the actor-critic architecture results in effective learning of the manipulation task under occlusions. Second, state-of-the-art reinforcement learning methods failed to learn the task of object pushing in the presence of even simple distractors, that occasionally occlude the object. Initializing the actor-critic network weights from policies learned in environments without distractors, quickly and effectively raises success rate in the difficult environment. This highlights the importance of curriculum learning regarding environment difficulty.

Our resulting hand-eye control policies outperform manipulation policies learned with a static camera. In summary our contributions are as follows:

- We introduce the problem of learning manipulation policies under occlusions, and propose agents that can control both hand and eye movement, to coordinate active perception and action, in environments with various types of distractors.

- We present modular actor-critic network architectures for action and perception in which only part of the state is exposed to the gripper controller, and where object detector modules are used to localize the object in the selected camera viewpoints. The proposed modular architectures outperforms non-modular alternatives.

Our code will be publicly available at https://github.com/ricsonc/ActiveVisionManipulation.

2   Related work

State estimation and reinforcement learning   Many Reinforcement Learning (RL) control methods manually design the state representation, that is often comprised of object and gripper 3D locations and poses, velocities, etc. which they assume given [2], or easily obtainable with an object detector [3, 4]. This often assumes there is no severe occlusion between the manipulated objects, the gripper or other distractors present in the scene. Tobin et al. [5] use synthetic data augmentation to learn detectors in cluttered environments. However, objects are almost never fully occluded in their setup, so is it not necessary for the camera to be active. Ebert et al. [6] learns a frame prediction model that can handle occlusions by using unoccluded past views to move the objects from. Instead, we take the complementary approach of keeping the object of interest in view using an active camera.

Other works attempt control directly from RGB images [4]. It has been shown that such frame-centric representations do not generalize under environmental variations between training and test conditions [7]. Our manipulation policy takes as input an RGB image, but also separately estimates the pose of objects in the scene using an object detector. We control the camera motion to facilitate the work of the object detector module, under occlusions from the environment.

Active vision   An active vision system is one that can manipulate the viewpoint of the camera(s) in order to investigate the environment and get better information from it [8, 9]. Active visual recognition attracted attention as early as 1980's. The work of [10] suggested many problems such as shape from contour and structure from motion, while ambiguous for a passive observer, they can easily be solved for an active one [11, 12]. Active control of the camera view point also helps in focusing computational resources on the relevant element of the scene [13]. The importance of active data collection for sensorimotor coordination has been highlighted in the famous experiment of Held and Hein in 1963 [14] involving two kittens engaged soon after birth to move and see in a carousel apparatus. The active kitten that could move freely was able to develop healthy visual perception, but the passively moved around kitten suffered significant deficits. Despite its importance, active vision was put aside potentially due to the slow development of autonomous robotics agents, necessary for the realization of its full potential. Recently, the active vision paradigm has been resurrected by approaches that perform selective attention [15] and selective image glimpse processing in order to accelerate object localization and recognition in static images [16, 17, 18]. Calli et al. [19] select camera views in order to maximize the success rate of object classification. Fewer methods have addressed camera view selection for a moving observer [20, 21, 22], e.g., for navigation [23, 24] or for object 3D reconstruction in synthetic scenes [25]. Radmard et. al [26] explicitly maps out occluded space in order to determine how to move the camera. Our approach learns such camera control guided by a reward of achieving a manipulation task, as opposed to manually designing such policy. Gualtieri and Platt [27] consider a robot that can focus on different regions of a precaptured pointcloud of the entire scene while performing a manipulation task. We do not assume pointcloud input, instead we learn to control the camera directly from RGB input. To the best of our knowledge, our work is the first (in recent years) to consider **active perception for action, as opposed to recognition.**

3   Reinforcement learning hand/eye coordination for object manipulation

We consider artificial agents that can control both their camera and gripper to complete manipulation tasks. We specifically investigate pushing objects to particular locations in the workspace in the presence of distractors, that may occlude occasionally the object of interest. Our proposed architectures can be extended to other manipulation tasks.

We consider a multi-goal Partially Observable Markov Decision Process (POMDP), represented by observations $o \in \mathcal{O}$, states $s \in \mathcal{S}$, goals $g \in \mathcal{G}$, hand (gripper) actions $a_g \in \mathcal{A}^{\mathcal{G}}$, and eye (camera) actions $a_c \in \mathcal{A}^{\mathcal{C}}$. Let $\mathcal{A} : \mathcal{A}^{\mathcal{C}} \times \mathcal{A}^{\mathcal{G}}$ denote the full action space. At the beginning of each episode, an observation-goal pair is sampled from the initial observation distribution $p(o_0, g)$. Each goal corresponds to a reward function $r_g : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$. At each timestep of an episode, the agent gets as input the current observation and goal, and chooses actions according to policy $\pi : \mathcal{O} \times \mathcal{G} \to \mathcal{A}$ which results in reward $r_t = r_g(o_t, a_t)$. The objective of the agent is to maximize the expected discounted future reward.

Our hand-eye controller architectures are represented by (generalized) modular actor-critic networks. We explored two distinct actor architectures, illustrated in Figure 1 (a), (b), both of which we couple with the same critic architecture, illustrated in Figure 1 (c).

The critic network takes as input an observation, namely, an RGB frame I_t, and encodes it using a what-where decomposition. Specifically, the input observation is encoded into:

- a low-dimensional embedding vector $f_t \in \mathbb{R}^{64}$ that represents the frame appearance. We use a convolutional neural network (CNN) depicted as a purple trapezoid in Figure 1 (c).
- a 3D centroid for the object of interest $\hat{o}_t \in \mathbb{R}^3$. A visual detector module [28] outputs a 2D bounding box for the object of interest. We recover the 3D location of the object by considering the camera ray which passes through the center of the detected 2D bounding box, and intersecting it with the horizontal plane on which the object lies, using ground-truth camera intrinsic and extrinsic parameters.

The predicted object location $\hat{o}_t$ is fed alongside the gripper location h_t (assumed known) and the desired goal object location g to the critic, alongside gripper and camera actions a_{g_t}, a_{c_t}, and the RGB image embedding f_t. Our (state, goal) representation thus reads:

$$(s_t, g) = ([f_t; \hat{o}_t, h_t], g), \tag{1}$$

where $\hat{o}_t, h_t, g$ stand for absolute 3D locations in the workspace. Empirically, we have found an object-centric representation for the object and gripper locations to lead to faster learning, namely:

$$(s_t, g) = ([f_t; h_t - \hat{o}_t], g - \hat{o}_t). \tag{2}$$

Such object-centric representation exploits translation invariance of the physical laws, and thus of the resulting policies, and has been used in previous works [2]. This invariance is mostly true in our environment as long as we do not go beyond the limits of the gripper. We use this object-centric representation in all our experiments.

A moving camera drastically increases the amount of variability exhibited in the input observation sequences, making it harder to learn manipulation policies directly from full frame input. We experimented with actor architectures which do not use the frame appearance embedding as input to the gripper actor policy, rather only the location of the object of interest. Specifically, we explore the following two actor architectures:

1. *Full* state information available to the gripper actor network (Figure 1 (a)). Frame appearance embedding f_t, object location $\hat{o}_t$ and gripper location h_t, are provided to both hand and eye actor subnetworks.
2. State *abstraction* for the gripper actor network (Figure 1 (b)). The image embedding f_t is provided as input only to the eye control policy, while the object and gripper locations are provided as input to both (hand and eye) controllers. In this way, the gripper actor network uses a manual yet way less variable state representation, which may be simpler to learn from.

Our hand/eye controllers are trained with reinforcement learning and Hindsight Experience Replay [2] (HER). HER introduced the powerful idea that failed executions – episodes that do not achieve

the desired goal – achieve some certain goal, and it is useful to book-keep such failed experience as successful experience for that alternative goal in the experience buffer. Such multi-goal experience is used to train a generalized policy, where actor and critic networks take as input both the current state as well as the current goal (as opposed to the state alone). Thanks to the smoothness of actor and critic networks, achieved goals that are nearby the desired goal, are used to learn useful value and actions, instead of being discarded. Training is carried out with off-policy deep deterministic policy gradients (DDPG) [29]. The agent maintains actor $\pi : \mathcal{S} \rightarrow \mathcal{A}$ and action-value (critic) $Q : \mathcal{S} \times \mathcal{G} \times \mathcal{A} \rightarrow \mathbb{R}$ function approximators. The actor is learned by taking gradients with loss function $\mathcal{L}_a = -\mathbb{E}_s Q(s, g, \pi(s, g))$ and the critic minimizes TD-error using TD-target $y_t = r_t + \gamma Q(s_{t+1}, g, \pi(s_{t+1}, g))$, where γ the reward discount factor. Exploration is carried out by adding ϵ normal stochastic noise to actions predicted by current policy. HER has shown to outperform *single goal* policy learning with deep deterministic policy gradients (DDPG) [29], and is currently a state-of-the-art RL method for continuous control. Since our goal representation is desired locations in the workspace the object should reach, HER is particularly well suited for our setup, since we do expect learned policies to change smoothly with respect to goal locations.
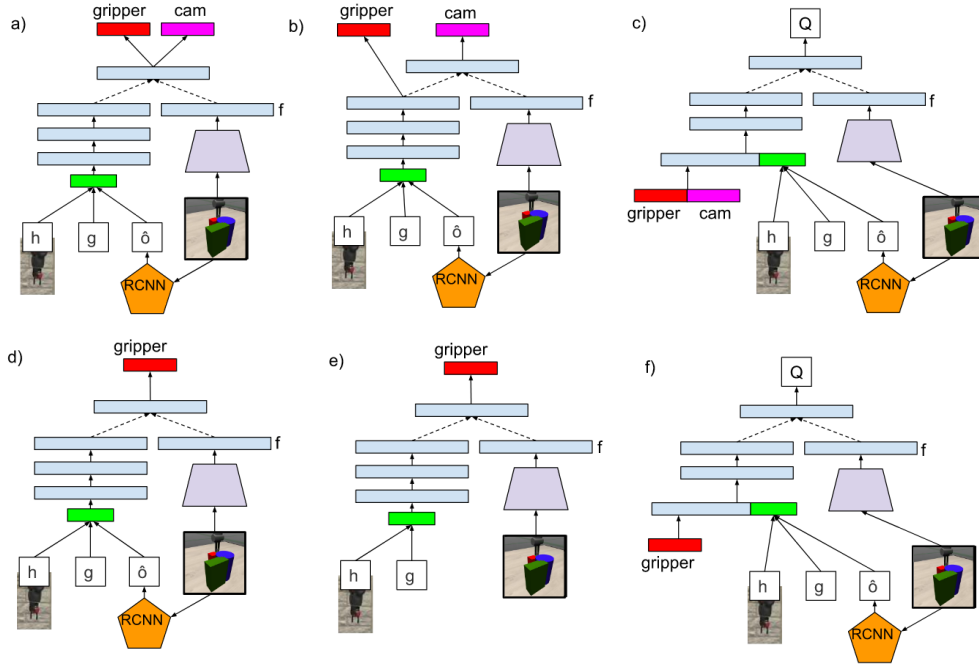

Figure 1: **Actors and critic architectures for active** *(top row)* **and static** *(bottom row)* **camera agents**. a) Actor network for *cam-active-full*. The same state representation is fed as input to hand and eye actor networks. b) Actor network for *cam-active-abstr*. The gripper actor responsible for pushing the object does not receive direct information from the RGB input, but only the 3D location of the object. c) Critic network for all active camera agents. d) Actor network for *cam-static*, the agent can only control its gripper, not its camera. e) Actor network for *cam-static-image*. The pretrained object detector subnetwork is omitted. f) Critic network for *cam-static*. Dotted lines denote elementwise addition. Trapezoids denote convolutional neural sub-networks, and blue rectangles denote fully connected layers.

Implementation details The convolutional network used in the actor and critic networks and depicted as purple trapezoid in Figure 1 takes as input a $64 \times 64 \times 3$ image. The network has 3 layers with 16, 32, and 64 kernels of size 4×4 with strides 2, 2, and 4. This is followed by a global average pooling layer, then a final linear layer with 64 units is applied. Batch normalization is used after every convolution layer.

4

All the fully connected layers in the actor and critic networks have 64 units each. Layer normalization and relu activation are used. The action output layer uses tanh activation function. In the critic architecture, the action is concatenated with the object-centric state representation after one layer.

At test time, we use an RCNN [28] as the object detector with a ResNet-101 backbone. The RCNN detector module is initialized with weights obtained from training for the MS COCO object detection benchmark [30], and further finetuned to detect the object of interest in 10000 initial states of our environments. We use amodal 2D bounding boxes as ground-truth. This allows the RCNN to infer the position of an object even under heavy occlusion. We randomly initialized the camera position while collecting the training data. The weights of the object detector module are fixed during reinforcement learning.

Since it is computationally prohibitive to use the detector module during training, we substitute it with a simulated RCNN. The simulated RCNN detects the object with probability $p = 1 - \sqrt{\mathrm{occ}}$ where occ is the proportion of the object mask which is occluded. When the object is detected, we add random normal noise ($\mu = 0, \sigma = 0.1$) to the bounding box coordinates and return that as the output. We found no improvement from fine-tuning the final policy using the real RCNN detector module, which is thus used only at test time. A much lighter detector convolutional backbone could have been used instead.
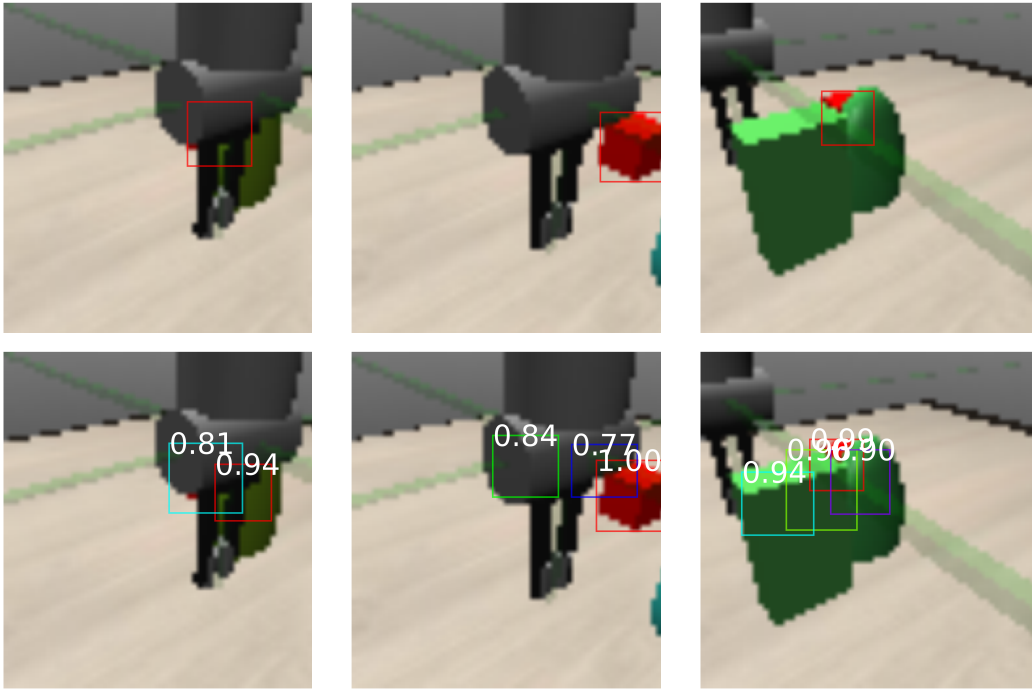

Figure 2: Sample predictions made by the trained RCNN. The RCNN is trained using *amodal* boxes. First row: ground truth, Second row: predictions, with confidence scores shown. Note that the RCNN is capable of inferring the position of the object even when it is completely occluded.

4   Experiments

We test the proposed method on a modified version of the `FetchPush-v0` environment from the OpenAI Gym using a Baxter robot. The goal of the agent is to push a object from its initial position to a goal position using its gripper. Both the object and the target are randomly initialized within a rectangular region. The agent controls motion of the gripper along a horizontal plane to the ground plane within a certain rectangular region, the gripper actions are thus dx, dy of the gripper displacement. An inverse kinematics solver is used to compute the required joint angles to achieve this displacement.

The agent can similarly move the camera along the horizontal plane, within a rectangular region. The camera can move up to 3 cm per time step and not more than 20 cm from the origin. We experimented with camera rotation in addition to just panning, but found that it did not help and was not used. There are randomly sampled distractors between the object and the camera. Each scene contains anywhere from 0 to 3 distractors. Each distractor is placed randomly within a rectangular region centered at the halfway point between the initial object location and the initial camera position. Each distractor is either a rectangular box, a truncated ellipsoid, or a cylinder. The color of the distractors is randomized. The camera is always initialized in the same starting location. The agent receives a -1 reward for every step it has not achieved the goal. When the object is within a tolerance radius 2 cm from the goal location, the episode ends with success. In all our results, we report the fraction of successful completions in 100 episodes in 3 training runs starting from different random seeds.

Our experiments aim to answer the following questions:

1. How do object occlusions affect reinforcement learning manipulation policies?
2. Can active vision help reinforcement learning of manipulation policies in the presence of occlusions?
3. Does object-centric attention help and to what extent?
4. Does curriculum learning in progressively more cluttered environments help policy learning under occlusions?
5. Are auxiliary visibility rewards necessary for learning active vision policies?

We compare the following architectures:

1. *cam-static*: an agent that can only control its gripper, not its camera. Its architecture is depicted in Figure 1 (d),(f).
2. *cam-static-image*: an agent that can only control its gripper, and its actor does not have an object detector module. Its architecture is depicted in Figure 1 (e), (f).
3. *cam-active-abstr*: an agent with hand/eye control and state abstraction. Its architecture is depicted in Figure 1 (b), (c).
4. *cam-active-full*: an agent with hand/eye control and without state abstraction. Its architecture is depicted in Figure 1 (a), (c).

Curriculum learning   Interestingly, **we were unable to learn a successful manipulation policy in the environments with distractor objects** under any architecture, starting from random weights —the detector submodule is excluded as it is always pretrained— as shown In Figure 3, right with the curve *cam-active-abstr (from scratch)*. When the actor-critic network weights are pretrained using RL on environments without distractors, then many of our architectural variants were able to learn successful policies in the environments with distractors, as shown in Figure 3. Therefore, all our experiments in environments with distractors use pretraining in environments without distractors. As see in Figure 3 left, an agent with an active camera (*cam-active-abstr*) is slower to train than an agent with a static camera (*cam-static*). For training active camera policies, we found that training was greatly accelerated by first pretraining with a static camera, which means, ignoring the predicted camera actions, this is illustrated in Figure 3, left, with the comparison between *cam-active-abstr* and *cam-active-abstr (pretrained)* curves.

Object-centric representation   Finally, we compare object-centric encoding of the state where the gripper and target 3D locations are provided relative to the 3D location of object of interest (Eq. 2) against an absolute state representation where gripper location, target location and object location are provided in absolute task space coordinates (Eq. 1). This comparison is illustrated in Figure 3, left, with the comparison between *cam-static (absolute)* and *cam-static* curves. Since we find that the object-centric representation performs better, we use it for all subsequent experiments.

Active vision for manipulation under occlusions   Though both active and static camera agents show similar success rates in the environments without distractors, (Figure 3 left), agents that can control their camera outperform agents with a static camera in environments with distractors, as shown in Figure 3, right, with the comparison between *cam-static* and *cam-active-abstr* curves.
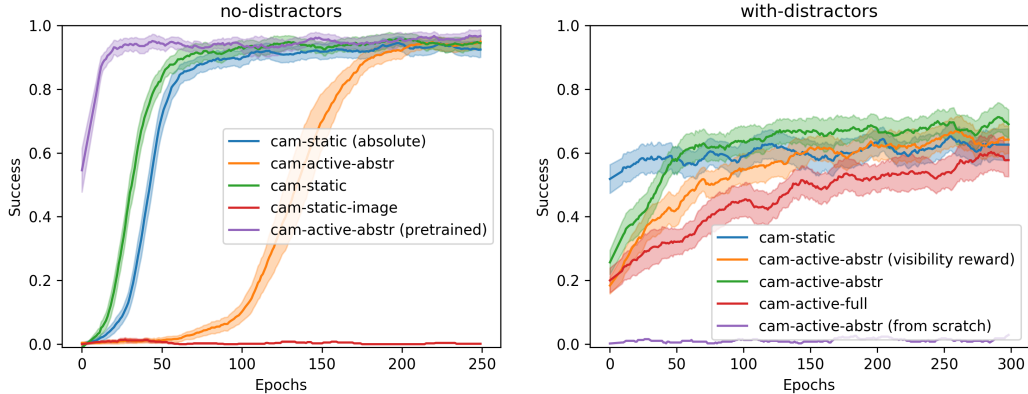
Figure 3: *Left:* **Environments without distractors.** Hand-eye policies train slower (*cam-active-abstr*), yet all architectures achieve good asymptotic performance. Hand-eye policies can be effectively pretrained from hand only policies (*cam-active-abstr* (pretrained)). Object-centric state encoding is beneficial (*cam-static* outperforms *cam-static (absolute)*). Finally, ignoring the location of the object of interest provided by the detector, and rather using only frame-centric appearance encoding does not result in successful behaviour (*cam-static-image*). *Right:* **Environments with distractors.** Active vision helps to handle occlusions from distractors (*cam-active-abstr* outperforms *cam-static*). State abstraction helps for the hand actor policy (*cam-active-abstr* outperforms *cam-active-full*). Training directly in the environment with distractors, without pretraining on the easier environment does not result in successful behaviours (*cam-active-abstr* (from scratch)). Auxiliary visibility reward is not necessary. Shaded area shows 1 standard error on the mean fraction of episodes which ended with success during training. We took the mean and computed the error over 3 training runs using different seeds.

The actor network that does not receive the frame appearance embedding learns faster than the variant that uses the frame appearance into the actor network. We visualize learned active vision policies in Figure 4. Video examples of the learned hand-eye coordination policies are available at https://github.com/ricsonc/ActiveVisionManipulation.

Auxiliary visibility reward We want to investigate whether auxiliary visibility rewards help to train active vision policies for manipulation. We compare the performance of an active camera controller trained under a combination of manipulation reward and an auxiliary visibility reward being 0.25 in value for every step that the object detector detects the object successfully, against an active camera controller trained solely from the manipulation reward. We expect the visibility reward to help the camera learn to move so that it increases visibility of the object of interest. In addition, the gripper may also aid the camera by pushing the object in a way which maximizes visibility. However, there is a potential downside that agent may favor actions less likely to push the object towards the target but rather improve visibility. We found such visibility reward did not help, as shown in Figure 3,right in the comparison between *cam-active-abstr* and *cam-active-abstr (visibility reward)*, as well as in the results of Table 1. This agrees with the observation from [2] that HER does not work well with dense rewards.

active	abstr	vis reward	success
yes	yes	no	0.663 ± 0.038
yes	yes	yes	0.574 ± 0.019
yes	no	no	0.523 ± 0.030
no	–	no	0.618 ± 0.008

Table 1: Success rate of the final policies at test time, averaged across 3 training runs using different random seeds in environments with distractors. Hand-eye control policies with state abstraction for the gripper actor and no visibility auxiliary reward perform best.
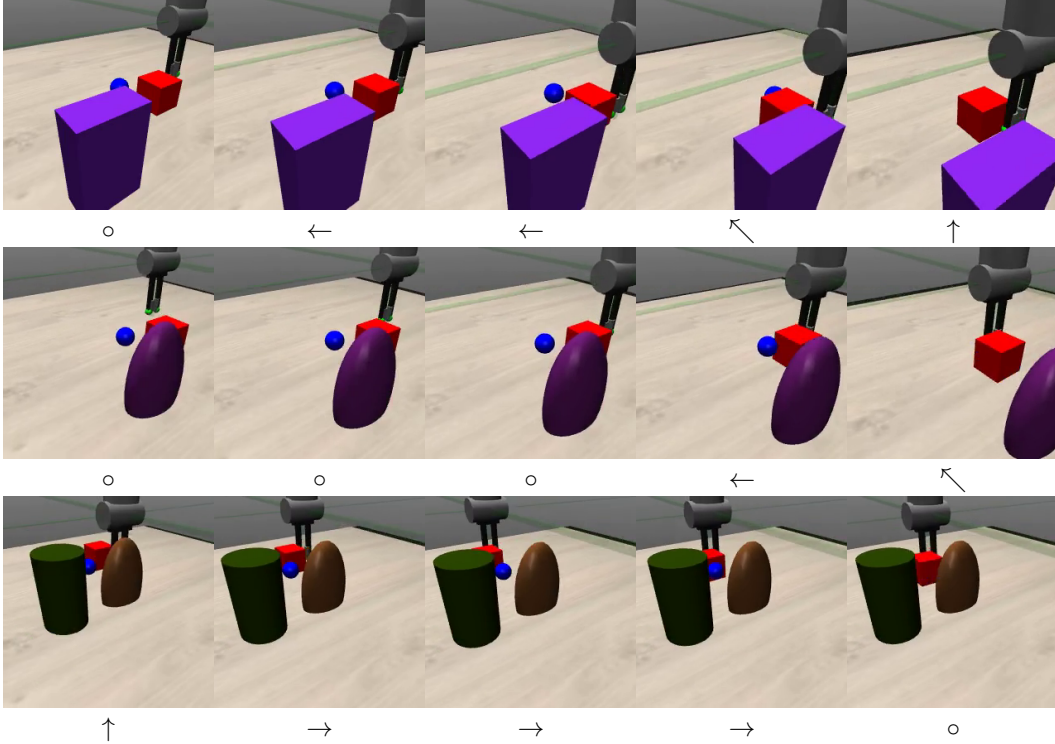
Figure 4: **Learned hand-eye control policies.** Each row corresponds to one episode, we show every other step of the episode. Since it is hard to tell the direction of camera movement from still frames, we draw arrows beneath each image showing the approximate direction of camera movement. In the top row, we see that the camera moves left and upwards to look over the obstacle. In the middle row, the robotic arm pushes the cube so that the left half is visible, and the camera moves left in order to expose the entire cube. In the bottom row, the cube is initially pushed leftward, so that if the camera is still, the cube would end up occluded by the cylinder. However, the camera moves right to compensate, so that the cube remains visible throughout the entire episode.

5   Discussion - Future Work

We proposed architectures for joint hand-eye coordination in the presence of environmental occlusions. We showed active camera control policies can effectively anticipate occlusions due to hand movement and move accordingly to aid state estimation. This work attempts to stimulate research interest towards vision for action, as opposed to vision for recognition. To the best of our knowledge, this is the first work that jointly considers the learning problems of active perception and manipulation.

Limitations of the current framework is that state estimation is memoryless, in other words, our object detectors operate independently at every frame. Future work includes combining active perception with visual memory architectures that encode a rich visual state that persist across timesteps of the episode, and integrates information of the visual scene in time. Another important and interesting direction is porting hand/eye coordination policies on a real robot for real time automated (un-instrumented) state estimation.

References

[1] S. Soatto. Actionable information in vision. In *ICCV*, 2009. 1

[2] M. Andrychowicz, D. Crow, A. Ray, J. Schneider, R. Fong, P. Welinder, B. McGrew, J. Tobin, O. P. Abbeel, and W. Zaremba. Hindsight experience replay. In *Advances in Neural Information Processing Systems*, pages 5055–5065, 2017. 2, 3, 7

[3] D. Xu, S. Nair, Y. Zhu, J. Gao, A. Garg, L. Fei-Fei, and S. Savarese. Neural task programming: Learning to generalize across hierarchical tasks. *CoRR*, abs/1710.01813, 2017. URL http://arxiv.org/abs/1710.01813. 2

[4] L. Pinto, M. Andrychowicz, P. Welinder, W. Zaremba, and P. Abbeel. Asymmetric actor critic for image-based robot learning. *CoRR*, abs/1710.06542, 2017. URL http://arxiv.org/abs/1710.06542. 2

[5] J. Tobin, R. Fong, A. Ray, J. Schneider, W. Zaremba, and P. Abbeel. Domain randomization for transferring deep neural networks from simulation to the real world. *CoRR*, abs/1703.06907, 2017. URL http://arxiv.org/abs/1703.06907. 2

[6] F. Ebert, C. Finn, A. X. Lee, and S. Levine. Self-supervised visual planning with temporal skip connections. *CoRR*, abs/1710.05268, 2017. URL http://arxiv.org/abs/1710.05268. 2

[7] K. Kansky, T. Silver, D. A. Mély, M. Eldawy, M. Lázaro-Gredilla, X. Lou, N. Dorfman, S. Sidor, D. S. Phoenix, and D. George. Schema networks: Zero-shot transfer with a generative causal model of intuitive physics. *CoRR*, abs/1706.04317, 2017. URL http://arxiv.org/abs/1706.04317. 2

[8] R. Bajcsy. Active perception. In *in Proceedings of the IEEE*, 1988. 2

[9] D. Wilkes and J. Tsotsos. Active object recognition. In *CVPR*, 1992. 2

[10] J. Aloimonos, I. Weiss, and A. Bandyopadhyay. Active vision. In *IJCV*, 1988. 2

[11] J. Denzler, M. Zobel, and H. Niemann. Information theoretic focal length selection for real-time active 3d object tracking. In *Proceedings Ninth IEEE International Conference on Computer Vision*, pages 400–407 vol.1, Oct 2003. doi:10.1109/ICCV.2003.1238372. 2

[12] E. Rivlin and H. Rotstein. Control of a camera for active vision: Foveal vision, smooth tracking and saccade. In *IJCV*, 2000. 2

[13] B. W. Tatler, M. M. Hayhoe, M. F. Land, and D. H. Ballard. Eye guidance in natural vision: Reinterpreting salience. *Journal of Vision*, 11(5):5, 2011. doi:10.1167/11.5.5. URL +http://dx.doi.org/10.1167/11.5.5. 2

[14] R. Held and A. Hein. Movement-produced stimulation in the development of visually guided behavior. In *Journal of Comparative and Physiological Psychology*, 1963. 2

[15] M. Ranzato. On learning where to look. In *arXiv preprint arXiv:1405.5488*, 2014. 2

[16] S. Mathe, A. Pirinen, and C. Sminchisescu. Reinforcement learning for visual object detection. In *CVPR*, 2016. 2

[17] A. Gonzalez-Garcia, A. Vezhnevets, and V. Ferrari. An active search strategy for efficient object class detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3022–3031, 2015. 2

[18] J. Caicedo and S. Lazebnik. Active object localization with deep reinforcement learning. In *ICCV*, 2015. 2

[19] B. Calli, W. Caarls, M. Wisse, and P. P. Jonker. Active vision via extremum seeking for robots in unstructured environments: Applications in object recognition and manipulation. *IEEE Transactions on Automation Science and Engineering*, pages 1–13, 2018. ISSN 1545-5955. doi:10.1109/TASE.2018.2807787. 2

[20] P. Ammirato, P. Poirson, E. Park, J. Kosecka, and A. C. Berg. A dataset for developing and benchmarking active vision. In *ICRA*, 2017. 2

[21] Y. Zhu, R. Mottaghi, E. Kolve, J. J. Lim, A. Gupta, L. Fei-Fei, and A. Farhadi. Target-driven visual navigation in indoor scenes using deep reinforcement learning. In *ICRA*, 2017. 2

[22] M. Malmir, K. Sikka, D. Forster, J. R. Movellan, and G. Cottrell. Deep q-learning for active recognition of germs. In *BMVC*, 2015. 2

[23] P. Mirowski, R. Pascanu, F. Viola, H. Soyer, A. J. Ballard, A. Banino, M. Denil, R. Goroshin, L. Sifre, K. Kavukcuoglu, et al. Learning to navigate in complex environments. *arXiv preprint arXiv:1611.03673*, 2016. 2

[24] S. Gupta, J. Davidson, S. Levine, R. Sukthankar, and J. Malik. Cognitive mapping and planning for visual navigation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2616–2625, 2017. 2

[25] D. Jayaraman and K. Grauman. Learning to look around. *CoRR*, abs/1709.00507, 2017. URL http://arxiv.org/abs/1709.00507. 2

[26] S. Radmard, D. Meger, J. J. Little, and E. A. Croft. Resolving occlusion in active visual target search of high-dimensional robotic systems. *IEEE Transactions on Robotics*, 34(3):616–629, June 2018. ISSN 1552-3098. doi:10.1109/TRO.2018.2796577. 2

[27] M. Gualtieri and R. Platt. Learning 6-DoF Grasping and Pick-Place Using Attention Focus. *ArXiv e-prints*, June 2018. 2

[28] S. Ren, K. He, R. B. Girshick, and J. Sun. Faster R-CNN: towards real-time object detection with region proposal networks. *CoRR*, abs/1506.01497, 2015. URL http://arxiv.org/abs/1506.01497. 3, 5

[29] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra. Continuous control with deep reinforcement learning. *CoRR*, abs/1509.02971, 2015. URL http://arxiv.org/abs/1509.02971. 4

[30] T. Lin, M. Maire, S. J. Belongie, L. D. Bourdev, R. B. Girshick, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. Microsoft COCO: common objects in context. *CoRR*, abs/1405.0312, 2014. URL http://arxiv.org/abs/1405.0312. 5