Learning visual servo policies via planner cloning

Ulrich Viereck*, Kate Saenko+, Robert Platt*
*Khoury College of Computer Sciences, Northeastern University; +Department of Computer Science, Boston University

***Abstract*—Learning control policies for visual servoing in novel environments is an important problem. However, standard model-free policy learning methods are slow to learn. This paper explores planner cloning: using behavior cloning to learn policies that mimic the behavior of a full-state motion planner in simulation. We propose Penalized Q Cloning (PQC), a new behavior cloning algorithm. We show that it outperforms several baselines and ablations on some challenging problems involving visual servoing in novel environments while avoiding obstacles. Finally, we demonstrate that these policies can be transferred effectively onto a real robotic platform, achieving approximately an 87% success rate both in simulation and on a real robot.**

I. Introduction

Visual servoing in novel environments is an important problem. Whereas classical approaches to the problem [3, 1, 15] make strong assumptions about the environment, deep learning methods solve these problems in more unstructured settings [17, 9, 14, 5, 13, 16, 7, 8]. Standard model-free reinforcement learning methods such as DQN [10], do poorly because they treat this as an unstructured model-free problem. Most current approaches to visual servoing in unstructured domains involve some form of imitation learning or behavior cloning [17, 9, 14, 5, 13, 16]. However, there are substantial differences between these methods and their relative performance is unclear.

This work focuses on a class of imitation learning approaches that we call *planner cloning*, where the expert is an approximately optimal motion planner that generates estimates of the optimal value function and optimal policy given full state feedback. This information is cloned onto a value function or policy over the observation space (i.e. camera images) that can be used at test time where full state feedback is unavailable.

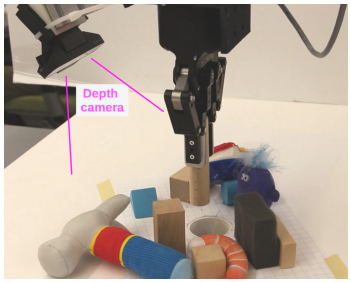


Fig. 1. One of the visual servoing test scenarios: Inserting a peg into a hole with collision avoidance.

The main contribution of this work is a new behavior cloning algorithm that we call *Penalized Q Cloning* (PQC). It has the following characteristics:

- It learns a value function using supervised value targets provided by the planner, similar to AGGREVATE [11].
- It incorporates a penalty into the value targets for suboptimal actions, similar to DQfD [2].
- For every experience, it updates the values for all feasible actions from that state, not just the action experienced.

This algorithm differs from AGGREVATE because it incorporates the value penalties and from DQfD because it uses supervised targets rather than TD targets. We compare PQC with several baselines and algorithm ablations and show that it outperforms all these variations on two challenging visual servoing problems in novel cluttered environments (i.e. Figure 1). The resulting policies achieve high success rates (approximately 87%) on challenging visual servo tasks.

II. Visual Servo Problem

Problem Statement: We assume we are given a discrete time system that includes a robotic end-effector, a camera, randomly placed clutter and and random object in gripper and target configuration. At the beginning of each episode the agent must move an object grasped into a desired pose relative to the environment. The agent must find a policy $\pi : Z \mapsto A$ over the observation space that minimizes the expected time to reach a goal state for the given system.

Train/test information asymmetry assumption: We assume that the agent has access to a fully modeled MDP and a simulator during training. During training, the simulator produces state and simulated camera observations. At test time, we assume that the agent does *not* observe the full state. Instead, the agent only has access to camera observations $z \in Z$.

III. Penalized Q Cloning (PQC)

In view of the train/test information asymmetry assumption, notice that there are really two relevant MDPs: 1) a fully modeled MDP over the underlying state space S; and 2) an unmodeled MDP over the observation space Z. We want to find a policy for the unmodeled MDP, but we can only plan in the modeled MDP. Our approach is therefore to use a full state motion planner to generate an "expert" policy and value function for the fully modeled MDP and then to project those solutions onto the unmodeled MDP using behavior cloning.

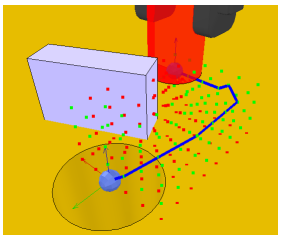


Fig. 2. Illustration of full state motion planning scenario.

A. Full state motion planner as the expert

First, we need to solve the fully modeled MDP for an expert value function and policy. In our particular case this

corresponds to a collision free motion planning problem. We use Djikstra's algorithm over a regular 3D grid of end-effector positions. Edges in the graph correspond to Euclidean distances plus a penalty for approaching obstacles. Shortest paths to goals in the graph correspond to optimal solutions to the modeled MDP. Solutions found by the planner provide to the behavior cloner: an approximately optimal policy π_E and an approximately optimal value function Q_E. Figure 2 shows an example of a trajectory found using this method.

B. Cloning a full state motion planner

We project these solutions generated by the planner onto the unmodeled MDP using behavior cloning. We do the following:
(a) Sample trajectories from expert: First, we roll out the expert π_E repeatedly. For each action, we simulate the resulting next state and query the expert for an approximate q-value, $Q_E(s, a)$. We store the sequence of experiences $(s_1, a_1, q_1), \ldots, (s_n, a_n, q_n)$ in a dataset $\mathbb{D}$, where $q_i = Q_E(s_i, a_i)$.
(b) Augment dataset with supervision for all actions: Because we have a small number of actions and the expert is implemented by a planner, we can generate supervision for all feasible actions from a given state, not just those executed by the simulator. Specifically, for each experience $(s, a, q) \in \mathbb{D}$, we generate $|A| - 1$ additional experiences $\{(s, a', Q_E(s, a'))|a' \in A \setminus a\}$ and add these to $\mathbb{D}$.
(c) Apply a penalty to non-expert actions: Unlike an approach like DAGGER [12] which clones the policy directly, here we are cloning the value function. This exposes us to a key failure mode: we may learn a good estimate of Q_E while still estimating π_E poorly. Under ideal conditions, our estimate Q is exactly equal to Q_E: $Q(s, a) = Q_E(s, a), \ \forall s, a \in S \times A$. In this case, the greedy policy of the learned value function is equal to expert policy: $\text{argmax}_a Q(s, a) = \text{argmax}_a Q_E(s, a) = \pi_E(s)$. However, since we are using a deep neural network to approximate Q_E, we can expect small errors. This is a problem because even small errors can result in a substantial divergence between $\text{argmax}_a Q(s, a)$ and $\pi_E(s)$. To combat this, we set the action values of non-expert actions to a fixed value c.

After generating the dataset $\mathbb{D}$ using the above, we use standard SGD-based methods to optimize

$$\mathcal{L}_{PQC} = \mathbb{E}_{(s,a,q)\sim\mathbb{D}}\Big[L(Q(s, a), q)\Big], \tag{1}$$

We call this *batch penalized Q cloning* (batch PQC, Alg. 1).
Relationship of this cloning method to prior work: This approach to behavior cloning draws elements from at least two different pieces of prior work. First, since we are cloning the value function rather than the policy, our method can be viewed as a form of AGGREVATE [11]. Second, the fact that we administer a penalty to non-expert actions is similar to what is done in DQfD [2] and ADET [6]. However, since both of those methods use TD learning, they must add an additional term into the loss function in order to achieve this. DQfD adds the relatively complex large margin loss term. ADET adds a cross entropy term between the policy implied by the value function and the expert policy. In contrast, since our method uses a fully supervised target, we can simply reduce the supervised target q value without the additional loss term. We experimentally compare our approach to AGGREVATE and DQfD in Section IV.

Algorithm 1 Batch Penalized Q Cloning (Batch PQC)

$\mathbb{D} \leftarrow \varnothing$
while more episodes to execute **do**
  $s_0 \sim \rho_0$
  for $t \in [0, T-1]$ **do** ▷ iterate over time steps
    $a_t = \pi_E(s_t)$
    for $\forall a \in A$ **do**
      if $a = a_t$ **then**
        $q \leftarrow Q_E(s_t, a)$
      else
        $q \leftarrow c$ ▷ apply penalty
      $\mathbb{D} \leftarrow \mathbb{D} \cup \{(s, a, q)\}$
    $s_{t+1} \leftarrow T(s_t, a_t)$
Find Q that minimizes $\mathcal{L}_{PQC}$
Return Q

C. Finetuning using TD learning

The cloning phase (pretraining) learns an approximately correct value function and the TD learning phase (finetuning using DQN in our case) makes small adjustments to improve performance (e.g. to improve upon a sub-optimal planner).

IV. Experiments in simulation

A. Experimental setup

We evaluate PQC against several algorithm variations and baselines on two visual servoing tasks: a peg insertion task and a block stacking task (Figures 4). In peg insertion, the robot starts execution with the peg in its hand and must move it until it reaches a goal pose just above the hole. The block stacking task is similar except that the robot starts with a block in its hand and must move it to a goal pose just above a second block. This is more challenging because it requires the policy to determine *which* block to stack upon.

B. Training details

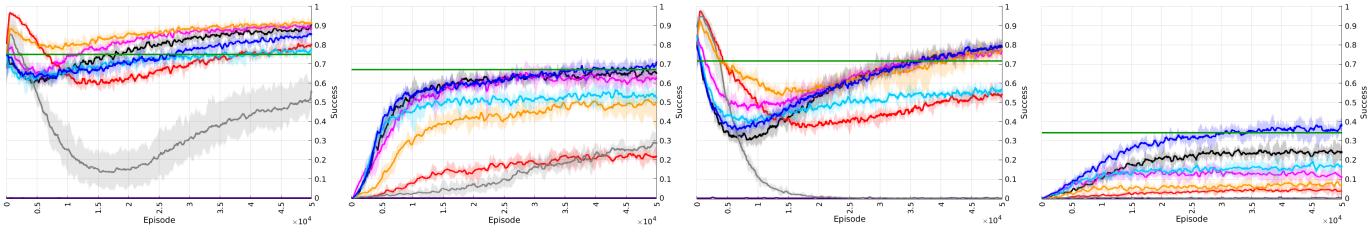
For each task, we created a dataset with 50k episodes by generating 500 scenes and rolling out 100 episodes per scene. Each scene was populated by random clutter and random peg/hole/block sizes and positions. For testing, we created a holdout dataset (50k episodes from 100 scenes).

C. Comparisons with ablations and baselines

Figure 3 compares the performance of fixed penalty batch PQC with a variety of ablations and baselines.

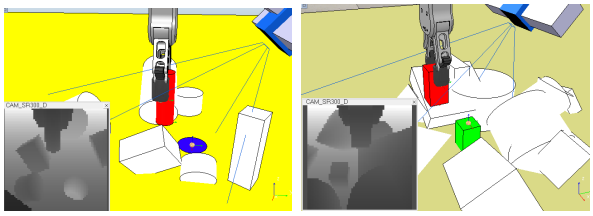
1. Batch PQC (green): Version of Batch PQC (Alg. 1).

2. Online PQC (blue): Same as batch PQC above except that it is trained online using a DAGGER-like rollout schedule.



(a) Peg-insertion; training set (b) Peg-insertion; holdout test set (c) Block-stacking; training set (d) Block-stacking; holdout test set

Fig. 3. Success rate as a function of training episode during cloning. (a,b) are peg-insertion. (c,d) are block-stacking. (a,c) show success rates on the training set. (b,d) show success rates on the holdout test set. Colors explained in Section IV-C. Results shown for training on 500 scenes. Note that PQQ (blue) significantly outperforms on the block stacking task on a holdout set (d). **Not shown:** The test success rate for batch PQC (blue) increases to about 87%(a) and 67%(b) if pretrained on 4k scenes and finetuned using TD learning. N**ote:** Batch PQC (green) shows the average success rate over all 50k rollouts after fully trained, and the online methods show success rates averaged over the last 500 rollouts.



(a) (b)

Fig. 4. (a) peg insertion scenario. (b) block stacking scenario.

		Failure Mode		
Scene	Success Rate	Collision	Not Recognize Hole	Missed Hole
1	14/20	2	3	1
2	19/20	-	1	-
3	17/20	1	2	-
4	18/20	-	1	1
5	18/20	-	-	2

TABLE I
RESULTS FROM PHYSICAL ROBOT TRIALS. SUCCESS RATES AND FAILURE MODES FOR EACH OF FIVE DIFFERENT SCENES.

3. Online PQC with no penalty (red): Ablation. Same as online PQC but without penalty for non-expert actions.

4. Online PQC one action update (cyan): Ablation. Same as online PQC except that only update the value of the action selected for execution.

5. Online PQC, relative penalty (orange): Same as online PQC except that we change line from: $q \leftarrow c$ to: $q \leftarrow Q_E(s_t, a) - l$, where $l = 0.2$.

6. DAGGER (black): Baseline. Classic DAGGER algorithm implemented using the standard cross entropy loss.

7. DQfD with DAGGER schedule (magenta): Baseline. This version of DQfD uses a single TD loss term plus supervised large margin classification loss (margin is 0.2).

8. DQN on-policy (purple): Baseline. DQN trained on-policy.

9. DQN with DAGGER schedule (grey): Baseline. DQN trained off-policy using the DAGGER schedule.

Based on these results, a few things are immediately clear. First, Figure 3 shows that on-policy DQN (purple) underperforms significantly. This justifies our fundamental choice to clone an expert rather than use model-free learning. Second, our proposed methods, batch PQC (green) and online PQC (blue) both perform similarly or slightly worse than two baselines, DQfD (magenta) and DAGGER (black), on the training set. However, they outperform these two baselines on the holdout test set. This is particularly true for the block stacking task which is harder than peg insertion because it requires observing the size of the grasped block in order to determine where to stack , see Figure 4-b. This suggests that PQC generalizes better to new scenes with different clutter configurations. Finally, the results indicate that the two ablations, online PQC with no penalty (red) and online PQC with one action update (the two ablations of online PQC), underperform, suggesting that both these elements of the algorithm are important.

V. VALIDATION ON A PHYSICAL ROBOT

We performed 100 proof-of-concept trials on the robotic system for the peg-insertion task on five novel scenes containing novel objects placed arbitrarily (example scene in Figure 1). We used a Robotiq 2F-85 gripper mounted on a UR5 arm in a tabletop setting. An Intel SR300 depth sensor was mounted near the robotic hand as illustrated in Figure 1. We train a pix2pix GAN [4] to learn a model that transforms a real image into an image that looks like what the simulator would have produced under similar circumstances. For more details, see longer version of this paper https://arxiv.org/pdf/2005.11810.pdf.

Table I shows the results of 86% success rate over all trials. Each trial began with the manipulator in a randomly selected pose (not in collision) continued until either reaching the hole, colliding with an object or timing out. We used the same policy for all trials – one that was trained in simulation using the full batch BQC cloning followed by TD learning (finetuning).

VI. CONCLUSION

We explore *planner cloning*, an approach that leverages the asymmetry in information that is available to the agent at train and test time. We generate an "expert" policy and q function in simulation with full state information and project these plans onto a policy that the agent can execute via behavior cloning. We propose Penalized Q Cloning (PQC)

that outperforms several algorithm ablations and baselines in simulation. Finally, we demonstrate that the resulting policies have similarly good performance on a real robotic system.

REFERENCES

[1] B. Espiau, F. Chaumette, and P. Rives. A new approach to visual servoing in robotics. *IEEE Transactions on Robotics and Automation*, 8(3):313–326, June 1992. doi: 10.1109/70.143350.

[2] Todd Hester, Matej Vecerik, Olivier Pietquin, Marc Lanctot, Tom Schaul, Bilal Piot, Dan Horgan, John Quan, Andrew Sendonaris, Ian Osband, et al. Deep q-learning from demonstrations. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[3] Seth Hutchinson, Gregory D Hager, and Peter I Corke. A tutorial on visual servo control. *IEEE transactions on robotics and automation*, 12(5):651–670, 1996.

[4] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1125–1134, 2017.

[5] Stephen James, Andrew J. Davison, and Edward Johns. Transferring end-to-end visuomotor control from simulation to real world for a multi-stage task. *CoRR*, abs/1707.02267, 2017. URL http://arxiv.org/abs/1707.02267.

[6] Aravind S Lakshminarayanan, Sherjil Ozair, and Yoshua Bengio. Reinforcement learning with few expert demonstrations. In *NIPS Workshop on Deep Learning for Action and Interaction*, volume 2016, 2016.

[7] Sergey Levine, Chelsea Finn, Trevor Darrell, and Pieter Abbeel. End-to-end training of deep visuomotor policies. *J. Mach. Learn. Res.*, 17(1):1334–1373, January 2016.

[8] Sergey Levine, Peter Pastor, Alex Krizhevsky, and Deirdre Quillen. Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection. In *Proc. International Symposium on Experimental Robotics (ISER)*, Tokyo, Japan, October 2016.

[9] Jeffrey Mahler and Ken Goldberg. Learning deep policies for robot bin picking by simulating robust grasping sequences. In *Conference on robot learning*, pages 515–524, 2017.

[10] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529, 2015.

[11] Stéphane Ross and J. Andrew Bagnell. Reinforcement and imitation learning via interactive no-regret learning. *CoRR*, abs/1406.5979, 2014. URL http://arxiv.org/abs/1406.5979.

[12] Stéphane Ross, Geoffrey Gordon, and Drew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 627–635, 2011.

[13] Fereshteh Sadeghi, Alexander Toshev, Eric Jang, and Sergey Levine. Sim2real view invariant visual servoing by recurrent control. *CoRR*, abs/1712.07642, 2017. URL http://arxiv.org/abs/1712.07642.

[14] Mengyuan Yan, Iuri Frosio, Stephen Tyree, and Jan Kautz. Sim-to-real transfer of accurate grasping with eye-in-hand observations and continuous control. *arXiv preprint arXiv:1712.03303*, 2017.

[15] Billibon H Yoshimi and Peter K Allen. Active, uncalibrated visual servoing. In *Proceedings of the 1994 IEEE International Conference on Robotics and Automation*, pages 156–161. IEEE, 1994.

[16] Fangyi Zhang, Jürgen Leitner, Michael Milford, and Peter Corke. Sim-to-real transfer of visuo-motor policies for reaching in clutter: Domain randomization and adaptation with modular networks. *CoRR*, abs/1709.05746, 2017. URL http://arxiv.org/abs/1709.05746.

[17] Yuke Zhu, Ziyu Wang, Josh Merel, Andrei Rusu, Tom Erez, Serkan Cabi, Saran Tunyasuvunakool, János Kramár, Raia Hadsell, Nando de Freitas, et al. Reinforcement and imitation learning for diverse visuomotor skills. *arXiv preprint arXiv:1802.09564*, 2018.