2019 International Conference on Robotics and Automation (ICRA)
Palais des congres de Montreal, Montreal, Canada, May 20-24, 2019

Robot eye-hand coordination learning by watching human demonstrations: a task function approximation approach

Jun Jin[1], Laura Petrich[1], Masood Dehghan[1], Zichen Zhang[1], Martin Jagersand[1]

***Abstract*— We present a robot eye-hand coordination learning method that can directly learn visual task specification by watching human demonstrations. Task specification is represented as a task function, which is learned using inverse reinforcement learning(IRL[1]) by inferring a reward model from state transitions. The learned reward model is then used as continuous feedbacks in an uncalibrated visual servoing(UVS[2]) controller designed for the execution phase. Our proposed method can directly learn from raw videos, which removes the need for hand-engineered task specification. Benefiting from the use of a traditional UVS controller, the training on real robot only happens at initial Jacobian estimation which takes an average of 4-7 seconds for a new task. Besides, the learned policy is independent from a particular robot, thus has the potential of fast adapting to other robot platforms. Various experiments were designed to show that, for a task with certain DOFs, our method can adapt to task/environment changes in target positions, backgrounds, illuminations, and occlusions.**

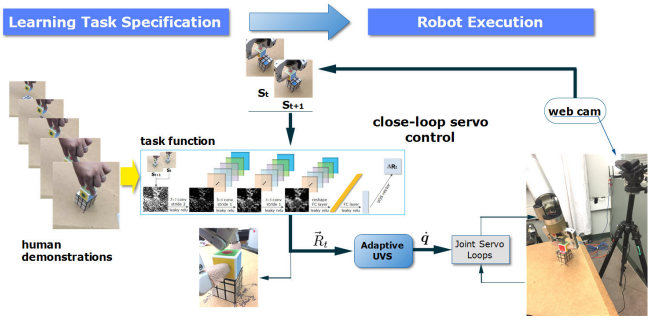

Fig. 1: Our proposed method firstly learns a task function from human demonstration videos. The learned task function receives real-time video streams and outputs a *reward vector* $\mathbf{R_t}$, which is used as continuous feedbacks in a close-loop UVS controller to guide robot motions in the execution phase.

I. Introduction

We address four problems in robot eye-hand coordination learning by watching. **1)** How to learn task specification from raw videos of human demonstration? **2)** How to directly train a visuomotor policy on a real robot with minimal hardware wear-out cost, while not using any simulators? **3)** How is the learned policy's generalization ability under different task settings (e.g., occlusions, illumination changes)? **4)** How to make the learned policy independent of this robot and be fast adaptive to a new robot platform?

Learning by watching human demonstrations provides a more intuitive way for task teaching. Recently proposed learning based methods (e.g., end-to-end [3], IRL [4], meta-learning [5]) provide strong generalization ability in different task settings, but training either needs simulators [6] or on real robot [7] [8], in which case, a certain time of training on this particular robot are needed, thus hardware wear-out and safety issues arise.

Compared to learning based approaches, traditional control methods (e.g., visual servoing [9], [10]) have the advantage of both data and control efficiency [11], good interpretability, almost no hardware wear-out issue, and can be fast adaptive to any practical robot platforms. However, problems regarding cumbersome task specification [12], tracking [13] and robustness under variances [14], impede its real-world applications.

The above-mentioned two approaches can be viewed in one unified framework, which is inspired by human eye-hand coordination. Human infants learn eye-hand coordination by first watching demonstrations, and then fulfilling the task in an exploratory manner. [15]. It is worth mentioning that 1) at a cognitive level, the learning process associates movement patterns with outcomes [15], and 2) from a control perspective, the human vision system acts as feedbacks [16] in a closed loop motion control manner.

How to relate visually observed outcomes to a motion pattern? In traditional visual servoing, this is more about understanding how tasks are specified and represented. Hager [12] and Dodds [17] proposed a programmable task specification method where an error function (a.k.a., task functions) is composed by setting various combinations of basic geometric constraints (e.g. point-to-point) [18]. This task function represents visual outcomes as a vector. Later, a Jacobian [2] that relates this vector and motion changes (e.g., joint velocity) is calculated. As we mentioned before, constructing this task function also needs robust tracking on image features.

Is it possible to directly learn a task function from raw image sequence and then use the task function in a traditional controller? We propose a method (Fig. 1) that learns a task function from raw video inputs based on Inverse Reinforcement Learning (IRL, [1]). The *task function* utilizes a *reward vector* to measures the motion outcome observed in image space. It is subsequently used as real-time feedback in a traditional closed-loop visual servoing controller with a minimal (4-7 seconds) of real robot online training. Major contributions are:

- Hand-engineered task specification in visual servoing is removed by directly learning from raw video demon-

[1]Authors are with Department of Computing Science, University of Alberta, Edmonton AB., Canada, T6G 2E8 jjin5, laurapetrich, masood1, vincent.zhang, martin.jagersand @ualberta.ca

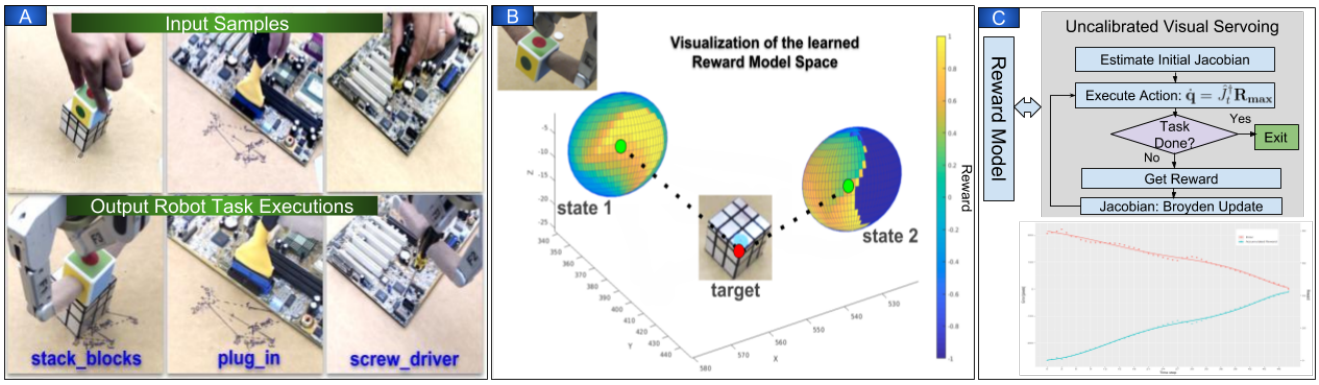

Fig. 2: A: Task definition. Given: videos of human demonstrations (above row). Question: Can robot do the same task? How about changing environments(occlusions, illumination changes)? From left to right, it represents a task with 1) regular geometric shape, 2) complex backgrounds and 3) small visual signatures respectively.**B: Visualization of the learned task function of stacking_blocks task.** The colored sphere represents the resulting reward when object moves from the center point (green dot) towards any direction on this sphere surface. The red dot defines the target. Left sphere and right sphere show results on different positions. Results show the region of directions going towards the target will return a **maximum** reward. **C: Execution phase** Up: The learned task function (reward model) acts as continuous feedback in an uncalibrated visual servoing controller.Down: An execution curve shows a proportional change pattern that, while the accumulated reward is increasing, error (in pixels) is minimized to zero. Errors are measured tediously by logging each step image and marking the pixel distance between object and target by hand.

stration;

- The use of a traditional control method makes directly training on real robot with minimal cost become possible, without using any simulators. It also provides fast adaptive ability to a different robot other than the trained one.
- Experimental design shows that for certain degrees of freedom (DOF) tasks, our method can generalize to variations in target positions, backgrounds, illuminations, and occlusions without prior retraining.
- Task interpretability in both the training and execution phase are provided by the learned *task function*;

II. Methods

A. Learning task function from raw videos of human demo

We propose **Incremental Maximum Entropy IRL (*InMaxEnt IRL*)** to learn the task function from demonstrations. Unlike most IRLs, our method learns from state transitions [19], [7], [20] which has the potential of bringing a better generalization ability [19]. Since it learns on state level, state changes can be incrementally stacked with increased number of demonstrations. For simplicity, we use one demonstration in this section to derive the basics of *InMaxEnt IRL*.

1) Terms and Definitions: Suppose we have an expert demonstration τ_i. Let s_t denotes the raw image state sampled at time t,and $\tau_i = \{s_0, ..., s_{n+1}\}$. Let's define the state change from s_t to s_{t+1} as ds_t^+, while the inverse direction from s_{t+1} to s_t as ds_t^-. In our experiments, ds_t^+ and ds_t^- are simply the result of modular subtraction between two images.

The outcome of the state changes is measured using a *Reward vector* : $\mathbf{R_t} \in \mathbb{R}^d$, where d is the task DOF. For simplicity, we assume each element in $\mathbf{R_t}$ is within range $[-1, 1]$. **Our goal** is to estimate a *task function* T:

$$\mathbf{R_t} = T(ds_t|\boldsymbol{\theta}) \tag{1}$$

, where ds_t defines a state change (e.g., ds_t^+ or ds_t^-), and $\boldsymbol{\theta}$ are parameters to learn. In this paper, a neural network (Fig. 3) is used to represent the task function T.

In order to have a scalar reward in IRL problem formation, the vector reward $\mathbf{R_t}$ is then converted to a scalar $R_t \in [-1, 1]$ by a dot product with $\mathbf{v} = \frac{1}{d}[1, ..., 1]^\top$.

Let's further define $\mathbf{R_t^+}$, R_t^+ as the observed state transition case and $\mathbf{R_t^-}$, R_t^- as the inverse direction case. They share the same task function with parameters $\boldsymbol{\theta}$ to estimate.

2) Generic action a_t: A generic action a_t^+ defines any actions can could deterministically cause state transition from s_t to s_{t+1}, whereas, a_t^- defines state change from s_{t+1} to s_t. So a generic action is independent of a specific robot. The mapping from generic actions to particular actions (e.g., joint velocity, torques) is done by a traditional controller in this paper. Global optimal control methods can also be used (e.g., RL[21]).

3) Boltzmann Distribution with Human Factor: A Boltzmann Distribution is commonly used in Maximum Entropy based IRL [22], which measures expert's preference over selection of trajectories. We borrow the same idea to measure expert's preference in selecting actions at s_t. Also, we argue that an expert is NOT selecting among all possible actions but selecting among high impact actions according to his/her confidence during demonstration.

For example, if an expert is pretty sure about what actions are optimal at s_t, he is selecting only from a small set of candidate actions. On the other hand, if he/she is not sure about what actions are 'good', the expert will have a hard time making decisions since the candidate action pool becomes large. We measure this behavior using a Boltzmann Distribution with Human Factor. This human

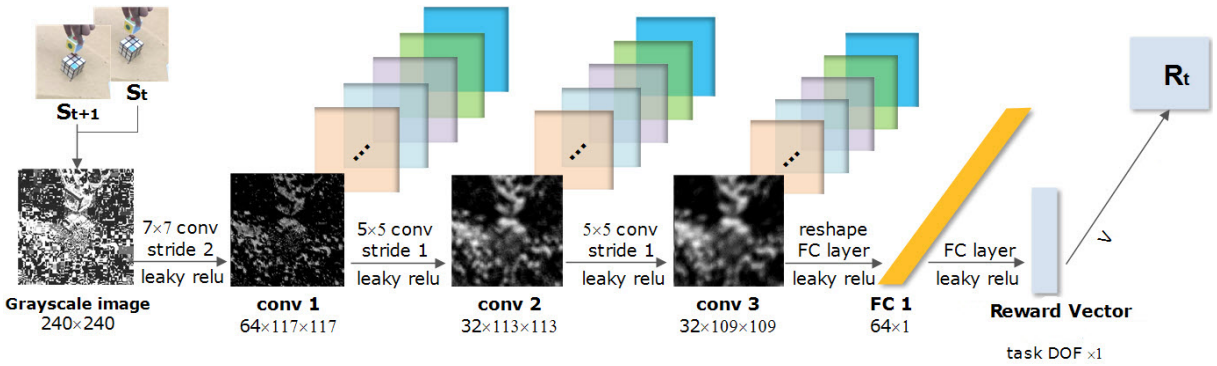

Fig. 3: A five-layer neural network is designed to represent *Task Function T*. It receives state change $ds_t{}^+$ computed by modular subtraction between s_{t+1} and s_t, while outputs a *Reward Vector* which measures the outcome when moving from s_t to s_{t+1}. For the purpose of maximum entropy problem formation, *Reward Vector* is then converted to a scalar *Reward* R_t by by a dot product with vector $\mathbf{v}$ (defined below), in which case, the *Task Function T* is a *reward function*.

factor refers to his/her confidence α during demonstration. The confidence level α will have a resulting variance σ_0, where high confidence level corresponds to low σ_0, vice versa.

Assume in state s_t, expert's action a_t^+ is selected from a set of possible actions $\mathcal{A}_t = \{a_{t1}, a_{t2}, ..., a_{tj}, ...\}$, where each action will return a reward R_{tj}. In order to emphasize high impact actions in this selection pool, we cast the possibility of a_{tj} appearing in the pool $\mathcal{A}_t$ as: $P(R_{tj}) = \mathcal{N}(R_t^+, \sigma_0^2)$. R_t^+ is the expert action's reward. So low reward actions will have lower probability appearing in this pool. We call this human factor prior.

As a result, we represent the probability of selecting action a_t at s_t as:

$$P(a_t|s_t) = \frac{1}{\mathcal{Z}_t} exp(R_t) P(R_t) \tag{2}$$

, where a_t defines a generic action (e.g, a_t^+ or a_t^-) $\mathcal{Z}_t = \int_{-1}^{1} exp(R_{tj}) P(R_{tj}) dR_{tj}$, is the partition function and difficult to estimate.

The role of human factor σ_0 casts how stronger the expert assumption is in IRL. It will determine how much effort should we invest in optimization. Obviously, if an expert is very confident in his/her demonstrations, we should invest greater efforts in optimization since we know it's **promising**. On the contrary, if an expert is very diffident in demonstration, we should invest less effort in optimization since the expert assumption is **questionable** now. The effect of using σ_0 will be further discussed later.

So for observed and the inverse sequence, the probability of selecting an observed an expert action a_t^+ and the inverse a_t^- at s_t is:

$$\begin{aligned} P(a_t^+|s_t) &= \frac{1}{\mathcal{Z}_t} exp(R_t^+) P(R_t^+) \\ P(a_t^-|s_t) &= \frac{1}{\mathcal{Z}_t} exp(R_t^-) P(R_t^-) \end{aligned} \tag{3}$$

4) Problem Formulation: Given an expert trajectory τ_i, we can formulate the problem by maximizing the likelihood of an observed state sequence $\{s_0, s_1, ..., s_{n+1}\}$ while minimizing the possibility of a negative sequence $\{s_{n+1}, ..., s_1, s_0\}$:

$$\boldsymbol{\theta}^* = \arg\max_{\boldsymbol{\theta}} \frac{P(s_0, s_1, ..., s_{n+1})}{P(s_{n+1}, ..., s_1, s_0)} \tag{4}$$

By applying the property of MDP, we have $P(s_0, ..., s_{n+1}) = P(s_0) \prod_{t=0}^{n} P(a_t^+|s_t)$. Unless prior knowledge is provided, $P(s_0)$, the initial prior, can be dropped in optimization. The same rule applies to negative sequence, and eqn. (4) can be written as:

$$\boldsymbol{\theta}^* = \arg\max_{\boldsymbol{\theta}} \prod_{t=0}^{n} \frac{P(a_t^+|s_t)}{P(a_{t+1}^-|s_{t+1})} \tag{5}$$

The above function is then expanded by adding the *negative log-posterior*. Combined with eq. (1), it's now a function of parameters $\boldsymbol{\theta}$ with input samples ds_t^+ and ds_t^-.

Computing the partition function $\mathcal{Z}_t$ is challenging. One possible solution is to use importance sampling with the assumption that $P(R_{tj}) \sim \mathcal{N}(R_t^+, \sigma_0^2)$, and then construct a generator network to approximate $\mathcal{Z}_t$. G plays the role as: given a reward R_t at state s_t, what's the corresponding state change ds_t. More accurate results could be obtained but the computation cost is expensive. We present a solution using *Boltzmann Factor* as a good trade-off between accuracy and computational cost.

5) Simplified Assumption: Boltzmann Factor: Is it possible to obliterate the partition function $\mathcal{Z}_t$ from numerator and denominator in eq. (5), since $P(a_t^+|s_t)$ and $P(a_t^-|s_t)$ share the same partition function $\mathcal{Z}_t$ at state s_t?

We borrow the concept of *Boltzmann Factor* to measure the relative entropy change between ds_t^+ and ds_t^-:

$$\begin{aligned} \boldsymbol{\beta_t} &= \frac{P(a_t^+|s_t)}{P(a_t^-|s_t)} \\ &= exp(R_t^+ - R_t^- + \frac{(R_t^+ - R_t^-)^2}{2\sigma_0^2}). \end{aligned} \tag{6}$$

Now the partition function $\mathcal{Z}_t$ is eliminated. Though not accurate, we can further assume that, the initial state transitions $P(a_0^+|s_0)$ and $P(a_{n-1}^-|s_{n+1})$ in the observed and negative

sequence respectively, can be dropped out in optimization. Now eq. (5) can be rewritten as:

$$\boldsymbol{\theta}^* = \arg\max_{\boldsymbol{\theta}} \prod_{t=1}^{n} \boldsymbol{\beta_t} \tag{7}$$

By applying the *log-posterior*, we have:

$$\boldsymbol{\theta}^* = \arg\max_{\boldsymbol{\theta}} \sum_{t=1}^{n} R_t^+ - R_t^- + \frac{(R_t^2 - R_t^-)^2}{2\sigma_0^2} \tag{8}$$

Since R_t is bounded to [-1,1], it's trivial to get the upper bound of the average cost value using σ_0^2 as:

$$upper\_bound = 2(1 + 1/\sigma_0^2) \tag{9}$$

Let's continue our discussion on human factor σ_0^2 in section A(3). Now the effect is more obvious. For higher confident expert demonstrations, the variance σ_0^2 will be smaller, thus have a higher upper bound. The learning process will push more towards this upper bound. For lower confidence demonstrations, vice versa. This is all about how stronger our expert assumption is in IRL.

This simplified version of *InMaxEnt IRL* using stochastic gradient ascend (Algorithm 1) is used in later experiments.

Algorithm 1: InMaxEnt IRL (using Boltzmann Factor)

Input: Expert demonstrations $\{\tau_1, ..., \tau_m\}$, task DOF number d, confidence level α
Result: Optimal weights $\boldsymbol{\theta}^*$ of Task Function T
Prepare State Change Samples $\mathcal{D}s^+$, $\mathcal{D}s^-$
for *i = 1:m* **do**
  for *t=1:sample size of* τ_i **do**
    Compute ds_t^+, $ds_t{}^-$
    Append samples: $\mathcal{D}s^+ \leftarrow ds_t^+$, $\mathcal{D}s^- \leftarrow ds_t{}^-$
  end
end
Compute σ_0 using α, construct $\mathbf{v}$ using d
Shuffle $\mathcal{D}s^+$, $\mathcal{D}s^-$; **Initialize** θ^0
for *n=1:N* **do**
  for *t=1:sample size* **do**
    Forward pass
    $R_t^+ = T(ds_t^+, \boldsymbol{\theta}) \cdot \mathbf{v}$, $R_t^- = T(ds_t^-, \boldsymbol{\theta}) \cdot \mathbf{v}$
    $ll(\boldsymbol{\theta}) = R_t^+ - R_t^- + \frac{(R_t^+ - R_t^-)^2}{2\sigma_0^2}$
    $err1 = \frac{\partial ll(\boldsymbol{\theta})}{\partial(R_t^+)} \cdot \mathbf{v}$
    $err2 = \frac{\partial ll(\boldsymbol{\theta})}{\partial(R_t^-)} \cdot \mathbf{v}$
    $grad1 = T.backProp(ds_t^+, \boldsymbol{\theta}^n, err1)$
    $grad2 = T.backProp(ds_t^-, \boldsymbol{\theta}^n, err2)$
    Gradient ascend update
    $\boldsymbol{\theta}^{n+1} = updateWeights(\boldsymbol{\theta}^n, grad1 + grad2)$
  end
end

B. Execution: map to a real robot using UVS

Now we have learned a reward model as the task function, uncalibrated visual servoing (UVS [2]) is used here to directly approximate an affine mapping from task space to robot joint space. As shown in Fig. 2C, UVS has four steps in a closed-loop control manner: 1) estimate an initial Jacobian by driving random motions and observing changes in task function. This is the only online training process on robot and cost an average of 4-7 seconds depending on hwardware speed. 2) Calculate action (joint velocity) using this Jacobian and execute this action. 3) Observe new changes in task function. 4) Broyden update the Jacobian.

Since we want an action that results in a maximum reward, which can be defined as $\mathbf{R_{max}} = [1, ..., 1]^\top \in \mathbb{R}^d$, where d is the task DOF. Action is computed by:

$$\dot{\mathbf{q}} = \hat{J}_t^\dagger \mathbf{R_{max}} \tag{10}$$

, where $\hat{J}_t^\dagger$ is the estimated Jacobian.

The biggest challenge is how to handle accumulated drifting error caused by continuous Broyden updates. In practice, we set a threshold vector $\mathbf{R_{thres}}$ to estimate $\hat{J}_{t+1}^\dagger$'s singularity proximity and perform $\hat{J}_0$ re-calibration if necessary.

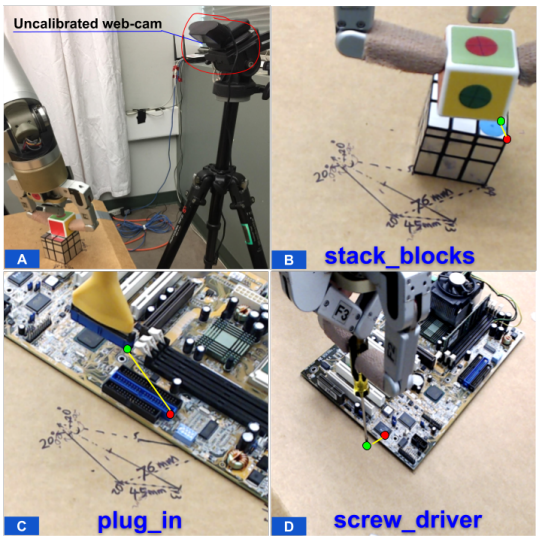

Fig. 4: A: Experimental setup. A low-cost uncalibrated webcam is used to record human demonstrations and guide robot motions. **B,C and D**: measurement of error using pixel distance between current position(green dot) to target(red dot). A threshold of 20 pixels ($\approx 7mm$) in a 580×580 image is used to determine a trial's success. Examples of successful and failure trials are shown in Fig. 5.

III. Experiments

Detailed experimental setup and evaluation metric is shown in Fig. 4. Our experimental objective aims to answer the following three questions: (1) What kind of task is our method capable of and what of task fails? (2) How does generalization differ under variances with tasks and environments? (3) How does performance change with varying number of demonstrations?

1) Capability with Different Tasks: We aim to evaluate four categories(Fig. 6) of fine manipulation tasks: (1) a 3DOF task with regular geometric shapes "stack blocks"; (2) a 3DOF task with complex backgrounds "plug in a socket on circuit board"; (3) a same "plug in" task with 6DOF. (4) a 3DOF task with small visual signatures "pointing with the

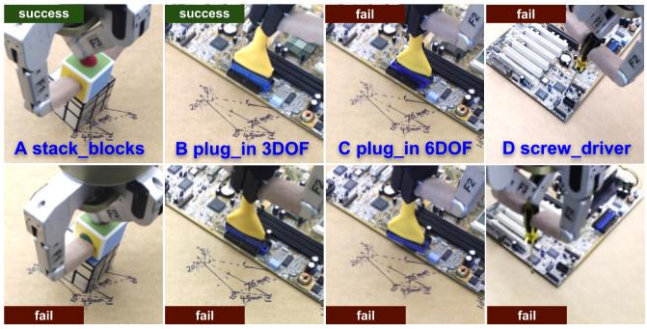

Fig. 5: Examples of successful and failure trials in each of the four tasks. A: stack_blocks task has a successful rate of **70%**. B: Plug_in 3DOF task is **60%**. The other two tasks all failed. The Plug_in 6DOF task failed since the learned task function is coarse and can't handle high DOF in the final UVS execution phase. The Screw_driver task failed since image changes between states are very small which cause larger errors in the learned task function.

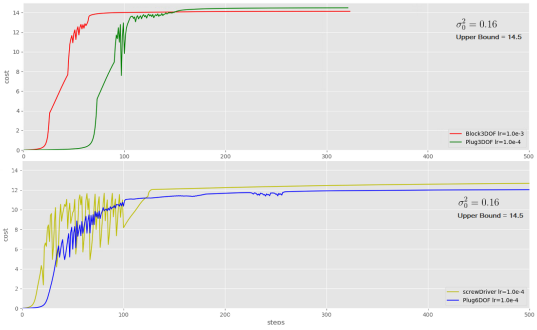

Fig. 6: Learning curves of all the 4 tasks in training the task function. All have the same variance $\sigma_0^2 = 0.16$ with corresponding upper bound 14.5 calculated in eq. (9). **Up:** Curves of stack_blocks task (red) and plug_in task (green). These two tasks have equal success rate and training pushes the cost very near to the upper bound. The resulting task function is more accurate. **Down:** Curves of plug_in 6DOF (blue) task and screw_driver (yellow) task. These two tasks all failed. It shows difficulty in training, and can't not push the cost near the upper bound. The resulting task function is more coarse thus causing failures.

TABLE I: Evaluation results on different tasks. Training on robots happens only in initial jacobian estimation. avg is 7 seconds which is minimum.

Task	Training (only images)	Successes	Mean error
stack_blocks	7.6 min	7/10	6.3±2.2
plug_in	9.8 min	6/10	6.3±1.6
screw_driver	13.3 min	0/10	-
plug_in 6DOF	11.84 min	0/10	-

tip of a screw driver". All tasks were trained on a moderate PC (one GTX 1080Ti GPU) using 11 human demonstrations.

Each task includes 10 trials. After each trial, visual error was manually measured (Fig. 4). Results are shown in Table I. Examples of success and fail trials of each task can be found in Fig. 5. Learning curves in training the task function, which show how cost is maximized are drawn in Fig. 6.

According to results showed in Table I and Fig. 6, the proposed method performs moderately well in tasks with regular geometric shapes and complex backgrounds, but fails in small visual change conditions and 6DOF tasks. In the screw_driver task, image changes are mostly caused by image noise instead of caused by actual robot motions. This also results in a longer training time. For the 6DOF task, as shown in Fig. 2B, the learned task function is still coarse and the adaptive UVS controller is a local method that's difficult to control with a coarse model.

2) Performance under Variances with Tasks and Environments: We tested the following settings (Fig. 7): (1) change_target1: The target block is translated 45mm long and rotated with 20°; (2) change_target2: The target block is translated 75mm long and rotated with 40°; (3) background1: Background is changed to a super messy one; (4) background2: Background is changed by adding an extra object; (5) object_occlusion: One face of the small block is occluded; (6) target_occlusion: One face of the target block is occluded; (7) change_illumination: An extra light source is placed directly opposite to the scene. Each of the seven setting had 10 trials. After each trial, visual error was manually measured following the same rule as stated before.

TABLE II: Evaluation results of generalization ability. Results show that it can generalize well under moderately changed target positions and backgrounds, occlusions and illumination changes.

Variant settings	Avg. steps	Successes	Error (Pixel)
(1) change_target1	23	5/10	7.6±3.8
(2) change_target2	22	2/10	2.0±0.3
(3) background1	45	4/10	9.7±2.6
(4) background2	24	9/10	6.6±4.0
(5) object_occlusion	26	7/10	12.1±2.2
(6) target_occlusion	36	6/10	9.0±4.2
(7) change_illumination	30	6/10	7.5±1.8

Results (Table II) show that it can generalize well under moderately changed target positions and backgrounds, occlusions and illumination changes. Since the state images are further processed using modular subtraction, it's not surprising that environment changes does not affect the performance as much, but the method performs poorly with large target or background changes.

3) Performance as Sample Size Increases: We are also interested in performance evaluation with varing numbers of human demonstrations. Using the same task "plug_in", we trained using 1, 5, and 11 human demonstrations, respectively. For each trained model, 10 trials were conducted and visual error was again used as a performance measure. Results are shown in Table III.

TABLE III: Evaluation results using different number of human demonstrations. Results show that the performance improves when using more demonstrations.

Demo Num	Training (only images)	Successes	Mean error
1	2.0 min	1/10	10.8±0
5	5.0 min	2/10	16.3±0.5
11	9.8 min	6/10	6.3±1.6

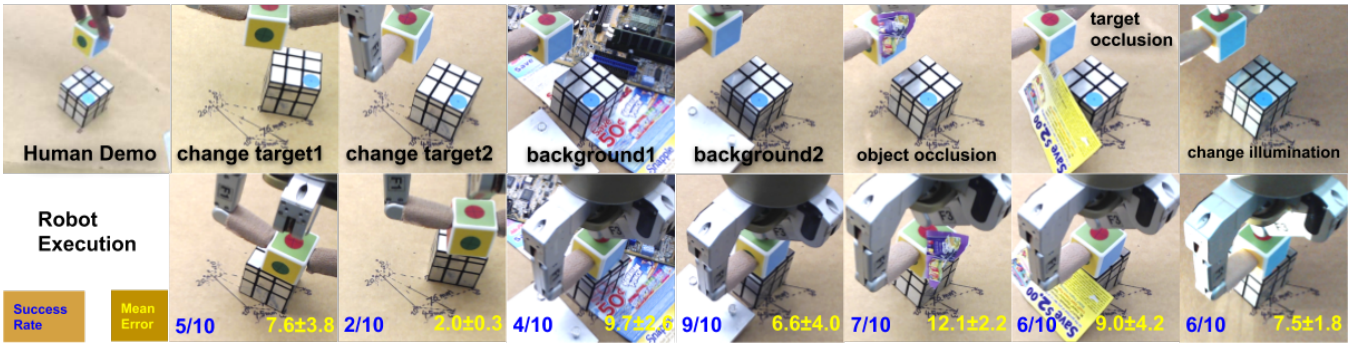

Fig. 7: Evaluation setup under different task/environment settings. Up row: initial settings; Down row: robot execution results, details are shown in Table II. Results show that it can generalize well under moderately changed target positions and backgrounds, occlusions and illumination changes.

4) Discussion on failures: Failure trials mainly come from two aspects: (i) the accuracy of task function, especially in cases with changing target position with patterns unseen in training samples. That's also the reason why increasing human demonstration numbers will improve its performance. (ii) the Jacobian estimation in UVS control since quality of $\hat{\mathbf{J}}_\mathbf{0}$ has a large effect on the control convergence. Jacobian estimation error mostly comes from the strategy of switching between Broyden update and $\hat{J}_0$ re-calibration, which is similar to the *exploration vs. exploitation* problem in Reinforcement Learning.

IV. Previous Works

Our work was inspired by research advances in visual servoing, including defining a task function using geometric constraints [12], [17], [13], [23], [24], direct visual servoing (DVS [25]) method to remove the tracking challenge, and increasing generalization ability in visual servoing [14] using deep neural networks. DVS is very similiar to our approach, however, it relies on a planar assumption and inconvenient task specification process.

Human demonstrations in place of manual task specification with end-to-end learning have been proposed to address these challenges [3]. IRL seeks to derive a *reward function* from observable actions and is closely related to learning from demonstration (also known as imitation learning or apprenticeship learning) [1]. This learned reward function is synonymous with the visual servoing error function, notwithstanding the scalar output of the reward function, whereas the error function outputs a vector with dimensionality determined by task DOF. Maximum entropy IRL was proposed to manage the problem of sub-optimal demonstrations [22], and Wulfmeier et al. represented the reward function using neural networks to handle non-linearity [26]. A challenge still exists in the estimation of the partition function Z, since it must solve the entire Markov Decision Process (MDP); this can be computationally expensive and unfeasible to generalize in a large action space or under unknown system dynamics. Finn et al. proposed an iterative solution using importance sampling to approximate a soft optimal policy[4] and recent works revealed the connection between IRL and generative adversarial networks (GAN) [27], [28]. Limitations of learning based approaches can be found in section I.

There are also other approaches on task specification learning, which are trying to build a task tree at the semantic level [29][30][31] and essential in practice to provide high level task programming.

V. Conclusions

We present a robot eye-hand coordination learning method that can directly learn task specification by watching raw demonstration videos, and map to joint velocity control using an adaptive UVS controller. *InMaxEnt IRL* is proposed to infer a task function from human demonstration videos. The use of a traditional controller enables efficient training on real robot with minimal hardware wear-out cost (4-7 seconds). It's also independent of a specific robot and in theory, provides fast adaptive ability on other robots.

Limitations and future directions: The major limitation of our method comes from its *local optimality*, not only derived from the reward based task function model, but also from the affine mapping in UVS control. Although it has this limitation, it provides the possibility to use global optimal control methods (e.g RL[21]) training directly on real-world robotic systems. Another limitation is the *accuracy of task function*, which is the main reason why higher DOF tasks failed. Besides, the assumption of Boltzman Factor can't work in non-prehensile tasks as $P(a_t^-|s_t$ is zero.

Future work could look at improving our simple network design to increase accuracy, for example through utilizing deeper networks, combing geometric invariant learning, and/or gaze selection[32]. Furthermore, with regards to task interpretability, it's still unclear how the *differential reward vector* is related to a high DOF task, and visual ambiguity from the single-camera view exists. The generalization to multiple view cameras could prove fruitful for future research.

Acknowledgement

The first author is grateful for receiving funding on his living expenses and tuition from China Scholarship Council (CSC) and the University of Alberta during his study.

REFERENCES

[1] P. Abbeel and A. Y. Ng, "Apprenticeship learning via inverse reinforcement learning," *Twenty-first international conference on Machine learning - ICML '04*, p. 1, 2004.

[2] M. Jagersand, O. Fuentes, and R. Nelson, "Experimental evaluation of uncalibrated visual servoing for precision manipulation," *Proceedings of International Conference on Robotics and Automation*, vol. 4, no. April, pp. 2874–2880, 1997.

[3] S. Levine, C. Finn, T. Darrell, and P. Abbeel, *The Journal of Machine Learning Research*, vol. 17, no. 1, pp. 1334–1373, 2016.

[4] C. Finn, S. Levine, and P. Abbeel, "Guided cost learning: Deep inverse optimal control via policy optimization," in *International Conference on Machine Learning*, 2016, pp. 49–58.

[5] T. Yu, C. Finn, A. Xie, S. Dasari, T. Zhang, P. Abbeel, and S. Levine, "One-shot imitation from observing humans via domain-adaptive meta-learning," *arXiv preprint arXiv:1802.01557*, 2018.

[6] S. James, A. J. Davison, and E. Johns, "Transferring end-to-end visuomotor control from simulation to real world for a multi-stage task," *arXiv preprint arXiv:1707.02267*, 2017.

[7] Y. Liu, A. Gupta, P. Abbeel, and S. Levine, "Imitation from observation: Learning to imitate behaviors from raw video via context translation," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2018, pp. 1118–1125.

[8] A. Xie, A. Singh, S. Levine, and C. Finn, "Few-shot goal inference for visuomotor learning and planning," *arXiv preprint arXiv:1810.00482*, 2018.

[9] G. J. Agin, *Servoing with visual feedback*. SRI International, 1977.

[10] F. Chaumette and S. Hutchinson, "Visual servo control. I. Basic approaches," *IEEE Robotics and Automation Magazine*, vol. 13, no. 4, pp. 82–90, 2006.

[11] P. I. Corke, "High-performance visual closed-loop robot control," Ph.D. dissertation, 1994.

[12] G. D. Hager and Z. Dodds, "On specifying and performing visual tasks with qualitative object models," *Proceedings-IEEE International Conference on Robotics and Automation*, vol. 1, no. April, pp. 636–643, 2000.

[13] M. Gridseth, O. Ramirez, C. P. Quintero, and M. Jagersand, "ViTa: Visual task specification interface for manipulation with uncalibrated visual servoing," *Proceedings - IEEE International Conference on Robotics and Automation*, vol. 2016-June, pp. 3434–3440, 2016.

[14] Q. Bateux, E. Marchand, J. Leitner, F. Chaumette, and P. Corke, "Training deep neural networks for visual servoing," in *ICRA 2018-IEEE International Conference on Robotics and Automation*, 2018, pp. 1–8.

[15] A. K. Rao, "Cognition and motor skills," in *Hand Function in the Child (Second Edition)*. Elsevier, 2006, pp. 101–113.

[16] C. Pehoski, "Object manipulation in infants and children," in *Hand Function in the Child (Second Edition)*. Elsevier, 2006, pp. 143–160.

[17] Z. Dodds, A. S. Morse, and N. Haven, "Task Specification and Monitoring for Uncali brat ed Hand / E ye Coordination *," no. May, 1999.

[18] Z. Dodds, M. Jägersand, and G. Hager, "A Hierarchical Architecture for Vision-Based Robotic Manipulation Tasks," *First Int. Conf. on Computer Vision Systems*, vol. 542, pp. 312–330, 1999.

[19] J. Fu, K. Luo, and S. Levine, "Learning robust rewards with adversarial inverse reinforcement learning," *arXiv preprint arXiv:1710.11248*, 2017.

[20] P. Sermanet, K. Xu, and S. Levine, "Unsupervised perceptual rewards for imitation learning," *arXiv preprint arXiv:1612.06699*, 2016.

[21] J. Peters and S. Schaal, "Reinforcement learning of motor skills with policy gradients," *Neural Networks*, vol. 21, no. 4, pp. 682–697, 2008.

[22] B. D. Ziebart, A. Maas, J. A. Bagnell, and A. K. Dey, "Maximum Entropy Inverse Reinforcement Learning," *AAAI Conference on Artificial Intelligence*, pp. 1433–1438, 2008.

[23] C. P. Quintero, M. Dehghan, O. Ramirez, M. H. Ang, and M. Jagersand, "Flexible virtual fixture interface for path specification in telemanipulation," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2017, pp. 5363–5368.

[24] M. Dehghan, Z. Zhang, M. Siam, J. Jin, L. Petrich, and M. Jagersand, "Online object and task learning via human robot interaction," *arXiv preprint arXiv:1809.08722*, 2018.

[25] G. Silveira and E. Malis, "Direct visual servoing: Vision-based estimation and control using only nonmetric information," *IEEE Transactions on Robotics*, vol. 28, no. 4, pp. 974–980, 2012.

[26] M. Wulfmeier, P. Ondruska, and I. Posner, "Maximum entropy deep inverse reinforcement learning," *arXiv preprint arXiv:1507.04888*, 2015.

[27] J. Ho and S. Ermon, "Generative adversarial imitation learning," pp. 4565–4573, 2016.

[28] C. Finn, P. Christiano, P. Abbeel, and S. Levine, "A connection between generative adversarial networks, inverse reinforcement learning, and energy-based models," *arXiv preprint arXiv:1611.03852*, 2016.

[29] R. Dillmann, "Teaching and learning of robot tasks via observation of human performance," *Robotics and Autonomous Systems*, vol. 47, no. 2-3, pp. 109–116, 2004.

[30] Y. Yang, Y. Li, C. Ferm\"{u}ller, and Y. Aloimonos, "Robot Learning Manipulation Action Plans by "Watching" Unconstrained Videos from the World Wide Web," *Twenty-Ninth AAAI Conference on Artificial Intelligence (AAAI-15)*, pp. 3686–3692, 2015.

[31] C. Xiong, N. Shukla, W. Xiong, and S. C. Zhu, "Robot learning with a spatial, temporal, and causal and-or graph," *Proceedings - IEEE International Conference on Robotics and Automation*, vol. 2016-June, pp. 2144–2151, 2016.

[32] R. S. Johansson, G. Westling, A. Bäckström, and J. R. Flanagan, "Eye-Hand Coordination in Object Manipulation," *Journal of Neuroscience*, vol. 21, no. 17, pp. 6917–6932, 2001.