Learning Torque Control for Visual Reacher Task

Amir Noohian[1] and Martin Jagersand[1]

***Abstract*— Reinforcement learning (RL) has emerged as a promising method for developing controllers for manipulators. Traditionally, RL designs for manipulators typically follow a position/velocity-based approach. In this paradigm, an RL policy generates target joint positions/velocities, which are then tracked by a proportional-derivative (PD) controller to generate joint torques. This study explores an alternative to the position/velocity-based RL paradigm by introducing a torque-based RL framework. In this framework, an RL policy directly predicts joint torques, eliminating the need for a PD controller. The effectiveness of the proposed torque control framework is demonstrated through a visual reacher task, where a robot endeavors to bring its end-effector close to a visual target. The results demonstrate that the robot successfully learns torque control for the visual reacher task.**

I. Introduction

Humans use hand-eye coordination to perform many day to day tasks including reaching for objects. In robotics, there has been significant research in this field. Hand-eye coordination tasks decompose into three components: perception which maps high dimensional image space to a lower visual feature space, control based on camera-robot geometry, and a method for task specification [1].

Visual servoing is one method used to perform these tasks without explicit modeling of the visuomotor function and the use of absolute world coordinate systems. Typically, in visual servoing, the robot uses visual video tracking to perform perception, and online Jacobian learning is used to learn how to control the robot, with tasks specified by visual selection. Generally, visuals servoing methods have low sample complexity, as the Jacobian can be initialized with a central difference method. However, the Jacobian is a locally linear approximation of the visuomotor function, so while is very accurate locally, but not necessarily optimal for the task globally [2].

In contrast, deep reinforcement learning (RL) has also been used to learn hand-eye coordination tasks in robotics. Neural networks are used to approximate an end to end policy that maps visual observations to actions. Usually, convolutional neural networks are used to extract features from the visual observations, while fully connected neural networks are used to map the features to actions. RL methods require more samples from the environment to learn compared to visual servoing methods, but they are more general in that the reward function is sufficient for task specification and can learn a close to optimal global motor policy for the task [3] [4].

[1]Department of Computing Science, University of Alberta, Edmonton, Canada. `[noohian, mj7]@ualberta.edu`

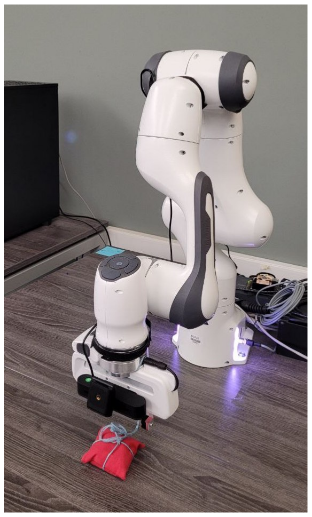

Fig. 1. The visual reacher task [8].

From a low-level control point of view, in visual servoing, usually the desired joint positions/velocities are sent to a PD controller to control the robot joints. The PD controller is treated as a low-level tracking module, and the PD gains are usually tuned manually to track desired positions/velocities well [5]. On the other hand, for RL-based methods, there are two main approaches in terms of the action space design: one is joint position/velocity, and the other is joint torque. When joint positions/velocity are used as actions, the learned policy outputs target joint angles which, similar to visual servoing, are then sent to a PD controller. When the action space is joint torque, actions are directly applied to motors to actuate a robot, without a PD controller [6] [7].

For manipulation tasks involving contacts with the environment, torque control is considered a safe option. In visual servoing, inverse dynamics control can achieve torque control but requires knowledge of robot dynamics parameters. [9]. However, in RL methods, joint torque control is achieved directly using joint torques as the action space [10].

Torque-based RL is highly valued for manipulation tasks for three key reasons. Firstly, unlike visual servoing, RL can achieve an optimal global policy. Secondly, torque control serves as a secure low-level control method for manipulation tasks where interaction with the environment is inevitable. Lastly, in contrast to visual servoing, which requires system parameters for torque control, RL does not rely on any prior knowledge of robot dynamics. In other words, the robot dynamics can be implicitly learned during the learning process, enabling the acquisition of nonlinearities such as

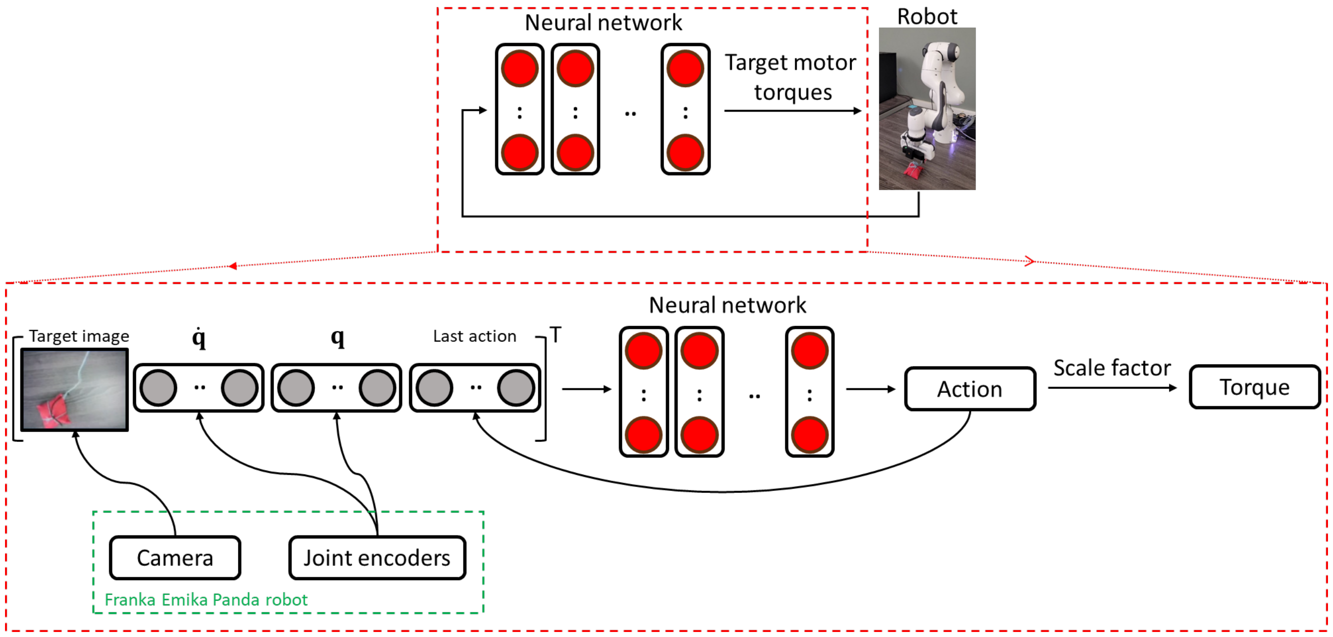

Fig. 2. The proposed learning torque control framework.

friction or backlash. In this study, our focus lies on mastering torque control for the visual reacher task, where a robotic manipulator endeavors to reach a visual target. We adopt the experimental setup described in [8] (Fig. 1).
The remainder of this report is structured as follows: Section II describes the end-to-end torque-based RL framework. In section III, the experimental setup and results are discussed. Finally, in section IV, we conclude the main key points of the study and suggest future works.

II. Learning Torque Control for Visual Reacher

In this section, we present our end-to-end learning torque control framework. Fig. 2 illustrates the observations and policy architecture used to predict actions.

A. Problem Formulation

We formulate the visual reacher task as a reinforcement learning problem where the objective is to learn a policy π that allows an agent to maximize its expected return for the given task. At each time step t, a state s_t is observed by the agent from the environment, and an action $a_t \sim \pi_t(s_t, a_t)$ is sampled from the policy π. The agent then applies this action and drives its current state to a new state s_{t+1} and a scalar reward $r_t = r(s_t, a_t, s_{t+1})$ is received. This process is iterated to generate a trajectory $\mathcal{D} = \{(s_0, a_0, r_0), (s_1, a_1, r_1), ...\}$. Therefore, the goal is to learn a policy that maximizes the agent's expected return $J(\pi)$ defined as

$$J(\pi) = \mathbb{E}_{\mathcal{D}\sim p(\mathcal{D}|\pi)}\left[\sum_{t=0}^{T-1} \gamma^t r_t\right], \quad (1)$$

where T is the time horizon for each episode, and $\gamma \in [0, 1]$ is a discount factor. $p(\mathcal{D}|\pi)$ represents the likelihood of a trajectory $\mathcal{D}$ under a policy π,

$$p(\mathcal{D}|\pi) = p(s_0) \prod_{t=0}^{T-1} p(s_{t+1}|s_t, a_t)\pi(a_t|s_t), \quad (2)$$

where $p(s_0)$ is the initial state distribution, and $p(s_{t+1}|s_t, a_t)$ denotes the state transition.

B. Observation Space

The observation space consists of four parts, which are robot joint positions and velocities, the last action, and the target image. Joint position, joint velocity, and last action are each 7-dimensional corresponding to the 7 joints of the robot. The last action is the last neural network output without being multiplied by the action scale. The target image is an RGB image captured by an RGB camera.

C. Action Space

The action space of our model is a 7×1 vector consisting of unscaled joint torques. This vector is then multiplied by a constant action scale to produce the actual desired torque values sent to the 7 motors. We augment the direct torque controller with a gravity compensation controller to improving training. This is a common technique [11] and is easily justified by the fact that many robot arms have built-in gravity compensation controllers that must be treated as part of the closed-loop dynamics. The resulting control law is

$$u = k_s \odot \pi(s) + g(q), \quad (3)$$

where u are the control torques, π is the policy, s is the state, k_s is a vector representing the scale factors, and $g(q)$ is the position dependant gravity compensation control term. The operator $\odot$ is used to indicate elementwise vector multiplication. We select the scale vector as $k_s =$

$[4.0, 4.0, 4.0, 4.0, 0.4, 0.4, 0.4]^{\mathrm{T}}$, taking into account that the first four joints of the robot exert higher torques compared to the last three joints. Finally, to guarantee the safety of the robot, all resulting torques are bounded within the ranges $[-80, 80]\mathrm{Nm}$ for the first four joints and $[-8, 8]\mathrm{Nm}$ for the last three joints.

D. Reward

We define a dense reward function as

$$R = 100R_{target} - 0.1R_{torque}. \tag{4}$$

R_{target} shows how big and close to the center the target is in the image and is defined as

$$R_{target} = \frac{1}{w \times h} \sum_{p \in \text{targetpixels}} (0.5 - |p_x|)(0.5 - |p_y|) \tag{5}$$

where w and h are image dimensions, and $p_x, p_y \in [-0.5, 0.5]$ are normalized coordinates of target pixels in the camera image. R_{torque} shows how large the joint torques are, which is defined as

$$R_{torque} = \sum_{\tau \in \text{jointtorques}} \tau^2 \tag{6}$$

where τ are the scaled joint torques. Overall, the total reward function encourages the agent to move towards the target, whilst satisfying smooth motions.

E. Policy Architecture and Training

The policy is modeled with a a convolutional encoder as the first layer followed by fully connected layers. The value function is modeled by a separate network with the same configuration as the policy network. The policy is optimized with soft actor-critic (SAC) algorithm [12]. The details of hyperparameters and implementations are publicly available on GitHub[1].

III. Experimental Setup and Results

We use a Franka-Emika Panda manipulator shown in Fig. 3 to evaluate our end-to-end learning torque control framework. The Franka-Emika Panda manipulator has 7 degrees of freedom (DoFs) with 7 actuated joints. We attach an RGB camera to the robot's wrist, which captures an 160×90 RGB image. A bean bag is randomly placed on the table, and the goal of the robot is to get close enough to the bean bag while it appears in the wrist camera image. The robot has the bean bag attached to its wrist by a string. This allows the robot to set the position of the bean bag randomly on the table at the beginning of each episode. The agent controls the robot joint torques (7 dimensions) at 25Hz. Each episode comprises 150 steps and requires approximately 6 seconds to complete. To ensure the safety of the robot, the position of the end-effector is confined within a box with dimensions of $[50, 70] \times [-10, 10] \times [25, 65]$ cm^3.

[1]https://github.com/amir-noohian/torque-visual-reacher-rl

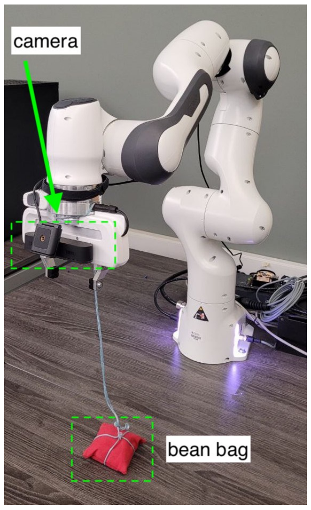

Fig. 3. The experimental setup for the visual reacher task [8].

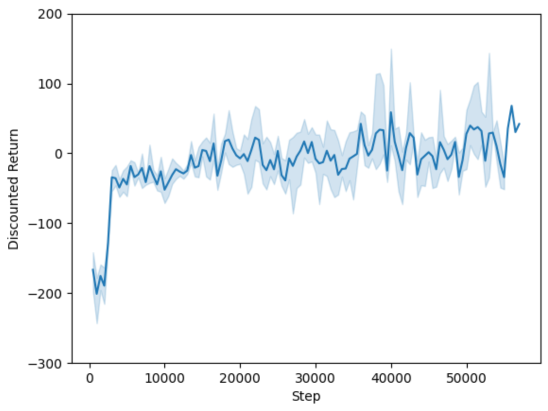

Fig. 4. The learning performance vs task steps for the visual reacher task. Shaded areas depict 95% confidence intervals.

Fig. 4 displays the average returns and confidence intervals over 3 runs. As depicted, there is a notable increase in returns during the initial learning steps, primarily due to the smooth motions the robot learns. Subsequently, the robot endeavors to maintain these smooth motions while reaching the bean bag. With the end-effector's motion confined within a safety bounding box, it can approach the bean bag up to 25 cm. Consequently, the value of R_{target} cannot reach the same magnitude as in cases where there is no safety bounding box. The experimental video can be found on Google Drive[2].

IV. Conclusion

In this paper, we investigated an alternative approach to learning the visual reacher task, employing joint torques as the action space in reinforcement learning, instead of joint positions/velocities. Our study included validation of

[2]https://drive.google.com/file/d/1fuuBm40w2ANwX1yltTYsQLN4HXKn5ZfQ/view?usp=sharing

the proposed method through a real-world visual-reacher task, where a manipulator attempts to reach a visual target. Results demonstrate that the torque-based visual reacher successfully reaches the visual target while maintaining smooth motion. The torque-based visual reacher serves as an initial experiment for torque-based RL and presents a promising avenue for future research and refinement. As an immediate experiment, we aim to evaluate the torque-based visual reacher with other reward functions, such as sparse and min-time. Furthermore, developing an intelligent safety module to adhere to the end-effector bounding box represents a promising approach to enhance the learning process, as it expands the robot's working space.

Acknowledgments

We would like to thank Professor Rupam Mahmood for his support during the course of this project, and Gautham Vasan and Fahim Shahriar for their help with the experimental setup. We would also like to thank Chen Jiang and Allie Luo for their help and feedback during the project.

References

[1] J. Jin, L. Petrich, M. Dehghan, Z. Zhang, and M. Jagersand, "Robot eye-hand coordination learning by watching human demonstrations: A task function approximation approach," in *2019 international conference on robotics and automation (icra)*, IEEE, 2019, pp. 6624–6630.

[2] M. Jagersand and R. Nelson, "Visual space task specification, planning and control," in *Proceedings of International Symposium on Computer Vision-ISCV*, IEEE, 1995, pp. 521–526.

[3] S. Levine, C. Finn, T. Darrell, and P. Abbeel, "End-to-end training of deep visuomotor policies," *Journal of Machine Learning Research*, vol. 17, no. 39, pp. 1–40, 2016.

[4] Y. Wang, G. Vasan, and A. R. Mahmood, "Real-time reinforcement learning for vision-based robotics utilizing local and remote computers," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2023, pp. 9435–9441.

[5] "Experimental evaluation of uncalibrated visual servoing for precision manipulation," in *Proceedings of International Conference on Robotics and Automation*, IEEE, vol. 4, 1997, pp. 2874–2880.

[6] P. Varin, L. Grossman, and S. Kuindersma, "A comparison of action spaces for learning manipulation tasks," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, 2019, pp. 6015–6021.

[7] S. Chen, B. Zhang, M. W. Mueller, A. Rai, and K. Sreenath, "Learning torque control for quadrupedal locomotion," in *2023 IEEE-RAS 22nd International Conference on Humanoid Robots (Humanoids)*, IEEE, 2023, pp. 1–8.

[8] A. Karimi, J. Jin, J. Luo, A. R. Mahmood, M. Jagersand, and S. Tosatto, "Dynamic decision frequency with continuous options," in *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, 2023, pp. 7545–7552.

[9] A. C. Leite, F. Lizarralde, and L. Hsu, "Hybrid adaptive vision—force control for robot manipulators interacting with unknown surfaces," *The international journal of robotics research*, vol. 28, no. 7, pp. 911–926, 2009.

[10] T. Haarnoja, V. Pong, A. Zhou, M. Dalal, P. Abbeel, and S. Levine, "Composable deep reinforcement learning for robotic manipulation," in *2018 IEEE international conference on robotics and automation (ICRA)*, IEEE, 2018, pp. 6244–6251.

[11] S. Gu, E. Holly, T. Lillicrap, and S. Levine, "Deep reinforcement learning for robotic manipulation with asynchronous off-policy updates," in *2017 IEEE international conference on robotics and automation (ICRA)*, IEEE, 2017, pp. 3389–3396.

[12] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," in *International conference on machine learning*, PMLR, 2018, pp. 1861–1870.