# Visual Servoing and Reinforcement Learning for Robotic Control

Steven Tang[1] and Martin Jagersand[1]

***Abstract*— This paper explores visual servoing and reinforcement learning (RL) for robotic control. In a simulated reaching task, we investigate the performance and sample efficiency of uncalibrated visual servoing and reinforcement learning. We analyze how different reward functions and representations affect reinforcement learning. Our experiments compare and contrast these approaches to provide insight into their strengths and weaknesses in hand eye coordination tasks, and explore how they can be combined to get the best of both methods.**

## 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.

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. [1] 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.

In contrast, deep reinforcement learning (RL) has also been used to learn hand-eye coordination tasks in robotics. [2] 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.

Visual servoing and reinforcement learning are two different methods that can be used to perform hand-eye coordination tasks. In this paper, we focus on visual servoing and reinforcement learning in the context of learning camera-robot geometry for robotic control. We use tracked points for perception and specify the task using the same error vector for both methods. We report quantitative performance data comparing the different methods in terms of sample complexity and final performance in a simulated reaching task. Moreover, we study how the two methods can be combined to get the best of both methods.

In Section II, we provide background information on visual servoing and reinforcement learning. In Section III, we discuss similar works in the literature. In Section IV, we describe the methods we explore in our experiments. In Section V, we describe our experimental setup and share our results. In Section VI, we conclude our work and discuss future work.

*We acknowledge the support of the Natural Sciences and Engineering Research Council of Canada (NSERC).

[1]Department of Computing Science, University of Alberta, Edmonton AB., Canada, T6G 2E8. {stang5, mj7}@ualberta.ca

## II. BACKGROUND

### A. Visual Servoing

Visual servoing is a robot control technique using vision. There are two main types of visual servoing. In calibrated visual servoing, the camera calibration parameters are known. [3] On the other hand, uncalibrated visual servoing relies on only on image features to control the robot, reducing calibration effort and is more general and independent of the specific experimental setup. Previous experiments have shown visual servoing to be an effective method for hand-eye coordination tasks. [4] We focus on uncalibrated visual servoing with two fixed cameras. In uncalibrated visual servoing, we seek to minimize the point to point error vector $\mathbf{f}(\mathbf{q})$ is a non-linear function of the joint angles $\mathbf{q}$.

$$\mathbf{f}(\mathbf{q}) = \begin{bmatrix} u_{g_1} - u_{e_1} \\ v_{g_1} - v_{e_1} \\ u_{g_2} - u_{e_2} \\ v_{g_2} - v_{e_2} \end{bmatrix} \tag{1}$$

Where $\mathbf{q} \in \mathbb{R}^d$ is the $d$-DOF robot's joint angles, $(u_{e_1}, v_{e_1}), (u_{e_2}, v_{e_2})$ are the image features of the end effector from the first and second cameras, and $(u_{g_1}, v_{g_1}), (u_{g_2}, v_{g_2})$ are the image features of the goal from the first and second cameras.

Given the Jacobian $\mathbf{J}$, we can use Newton's method to solve for the $\Delta\mathbf{q}$ that minimizes a linear approximation of the error function.

$$\mathbf{J}_\mathbf{f}(\mathbf{q})\Delta\mathbf{q} = -\mathbf{f}(\mathbf{q}) \tag{2}$$

The update to the joint angles $\Delta\mathbf{q}$ is given by

$$\Delta\mathbf{q} = -\mathbf{J}_\mathbf{f}(\mathbf{q})^+\mathbf{f}(\mathbf{q}) \tag{3}$$

Where $\mathbf{A}^+$ is the Moore-Penrose pseudoinverse of $\mathbf{A}$. The robot controller is then given the update to the joint angles

$\Delta\mathbf{q}$, and the joint angles are updated using the following equation.

$$\mathbf{q}_{t+1} = \mathbf{q}_t + \Delta\mathbf{q}_t \tag{4}$$

The Jacobian $\mathbf{J_f}(\mathbf{q})$ is initialized with the central difference approximation. We estimate each column independently, by perturbing one joint, keeping the other joints fixed.

$$\mathbf{J_f}(\mathbf{q})_i = \frac{\mathbf{f}(\mathbf{q} + \epsilon\mathbf{e}_i) - \mathbf{f}(\mathbf{q} - \epsilon\mathbf{e}_i)}{2\epsilon} \tag{5}$$

Where $\mathbf{J_f}(\mathbf{q})_i$ is $i$-th column of the Jacobian, the $\mathbf{e}_i$ is the $i$-th standard unit vector, using a small perturbation $\epsilon$, in our case, $0.1$ radians.

The Jacobian $\mathbf{J_f}(\mathbf{q})$ approximation can be updated online using Broyden's method.

$$\mathbf{J_f}(\mathbf{q})_{t+1} = \mathbf{J_f}(\mathbf{q})_t + \frac{\mathbf{f}(\mathbf{q})_t - \mathbf{J_f}(\mathbf{q})_t\Delta\mathbf{q}_t}{\Delta\mathbf{q}_t^\top\Delta\mathbf{q}_t}\Delta\mathbf{q}_t^\top \tag{6}$$

### B. Reinforcement Learning

In contrast, reinforcement learning learns from trial and error interaction with the environment. RL acts in a Markov Decision Process defined by the tuple $(\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma)$ where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, $\mathcal{P}$ is the transition probability function, $\mathcal{R}$ is the reward function, and $\gamma \in [0, 1]$ is the discount factor. The environment takes in an action and returns the next state and reward according to the transition probability function and reward function respectively.

The objective of the robot agent is to learn a policy $\pi(\mathbf{s}_t) = \mathbf{a}_t$ that maps states from the environment to actions which maximizes the expected return $R = \mathbb{E}[\sum_{t=0}^{T}\gamma^t r_t]$ where $r_t$ is the reward at time $t$, and $T$ is the time horizon. Often, deep neural networks are used to represent the policy. The policy is learned using gradient descent methods to maximize the expected return.

## III. RELATED WORKS

In *Model-based and model-free reinforcement learning for visual servoing* Farahmand et al., compares the performance of model-based/model-free RL based on the Regularized Fitted Q-Iteration algorithm, and uncalibrated visual servoing using only the first 3 DOF of the robot with discrete actions. [5] In our work, we compare the performance of uncalibrated visual servoing and reinforcement learning using 3, 4, and 7 DOF robots with continuous actions.

In *End-to-End Training of Deep Visuomotor Policies*, Levine et al. discuss end-to-end training of deep visuomotor policies, which uses a neural network architecture with a spatial softmax, and trains with a guided policy search, which combines supervised learning with a trajectory-centric RL algorithm that provides supervision on the policy. [2] They evaluated their technique on a range of real-world manipulation tasks.

In *Asynchronous Reinforcement Learning for Real-Time Control of Physical Robots*, Yuan and Mahmood demonstrate a system that learns to reach visual targets from pixels within 2 hours of experience using a real 5 DOF robot. [6] The paper's focus is on evaluating sequential and asynchronous learning, but serves as a good example of developing practical RL systems for real world visual robotic control.

## IV. METHODOLOGY

### A. Representation

In the uncalibrated visual servoing problem, we consider the representation of the visuomotor function to be the Jacobian which is learned using Broyden's method.

In reinforcement learning, we consider neural networks and a Neural Jacobian approach to represent the visuomotor function.

*1) Neural Networks:* Commonly used in reinforcement learning, neural networks are used to represent the policy.

In our case, we use fully connected neural networks parameterized by $\theta$ to approximate the end to end policy. We compare the performance of different neural network architectures, including the number of layers and the number of neurons per layer. We use the rectified linear unit, ReLU, activation function for hidden layers, and $\tanh$ for the output layer. We use the Adam optimizer with a learning rate of $0.001$.

For example, a 2-layer neural network with $l$ neurons per layer is given by

$$\pi_\theta(\mathbf{s}) = \tanh(\mathbf{W}_2\text{ReLU}(\mathbf{W}_1\mathbf{s} + \mathbf{b}_1) + \mathbf{b}_2) \tag{7}$$

Where $\mathbf{W}_2 \in \mathbb{R}^{d\times l}$, $\mathbf{W}_1 \in \mathbb{R}^{l\times d}$, $\mathbf{b}_2 \in \mathbb{R}^d$, $\mathbf{b}_1 \in \mathbb{R}^l$, and $\theta = \{\mathbf{W}_1, \mathbf{W}_2, \mathbf{b}_1, \mathbf{b}_2\}$ are the learned parameters of the neural network.

*2) Neural Jacobian:* Inspired by the Neural Jacobian approach from [7], rather than learning a fully connected neural network, we can use a neural network to approximate a Jacobian to model the visuomotor function. However, in contrast to the supervised learning treatment taken by Przystupa et al., we use reinforcement learning to learn the Neural Jacobian, which to our knowledge is a novel policy representation in reinforcement learning.

For example, a 2-layer neural network with $l$ neurons per layer that outputs a neural Jacobian is given by

$$\mathbf{J}_\theta(\mathbf{s}) = \mathbf{W}_2\text{ReLU}(\mathbf{W}_1\mathbf{s} + \mathbf{b}_1) + \mathbf{b}_2 \tag{8}$$

Where $\mathbf{W}_2 \in \mathbb{R}^{d\cdot n\times l}$, $\mathbf{W}_1 \in \mathbb{R}^{l\times d}$, $\mathbf{b}_2 \in \mathbb{R}^{d\cdot n}$, $\mathbf{b}_1 \in \mathbb{R}^l$, and $\theta = \{\mathbf{W}_1, \mathbf{W}_2, \mathbf{b}_1, \mathbf{b}_2\}$ are the learned parameters of the Neural Jacobian.

Similar to uncalibrated visual servoing, the Neural Jacobian is then used to compute the action update using the following equation

$$\pi_\theta(\mathbf{s}) = \tanh(\mathbf{J}_\theta^+(\mathbf{s})\mathbf{f}(\mathbf{q})) \tag{9}$$

### B. Methods

There are several methods that we use to learn the visuomotor function. In uncalibrated visual servoing, we use central differences to initially approximate the Jacobian. We can also use Broyden's method to update the Jacobian online. In our experiments, we use a constant Jacobian, as we found that the online updates did not noticeably improve performance.

*1) Twin Delayed DDPG (TD3):* There are numerous reinforcement learning algorithms. When compared to Soft Actor Critic (SAC) and Proximal Policy Optimization (PPO), we found TD3 to be the most sample efficient and stable algorithm for our task, we use TD3 in the following experiments. TD3 is an off-policy algorithm, it learns a Q-function in addition to the policy which is used to improve sample efficiency. TD3 is a variant of Deep Deterministic Policy Gradient (DDPG) that uses clipped double-Q learning, delayed policy updates, and target policy smoothing, which improves stability and performance. [8] We use RL algorithm implementations from `stable-baselines3` [9]. We use the default hyperparameters for TD3.

*2) Reward Function:* In reinforcement learning, the choice of the reward function is important. We compare the performance of several reward functions. We compare the sparse, timestep, and dense reward functions.

The sparse reward function is the simplest, it returns a reward of 1 if the end effector is within a threshold distance $\epsilon$ of the goal position, and 0 otherwise.

$$r_{\text{sparse}}(\mathbf{q}) = \begin{cases} 1 & \text{if } \|\mathbf{f}(\mathbf{q})\|_2 < \epsilon \\ 0 & \text{otherwise} \end{cases} \tag{10}$$

The timestep reward function is used to encourage the agent to complete the task as quickly as possible. It returns a reward of 0 if the end effector is within a threshold distance $\epsilon$ of the goal position, and -1 otherwise.

$$r_{\text{timestep}}(\mathbf{q}) = \begin{cases} 0 & \text{if } \|\mathbf{f}(\mathbf{q})\|_2 < \epsilon \\ -1 & \text{otherwise} \end{cases} \tag{11}$$

The dense reward function gives more shaped feedback to the agent, it returns the negative of the Euclidean distance between the end effector and the goal position.

$$r_{\text{dense}}(\mathbf{q}) = -\|\mathbf{f}(\mathbf{q})\|_2 \tag{12}$$

### C. Combined Approaches

We also explore two approaches that combine visual servoing and reinforcement learning, including Residual Reinforcement Learning and Jump Start Reinforcement Learning.

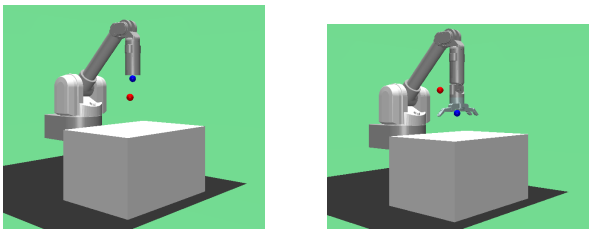
*1) Residual Reinforcement learning:* Residual Reinforcement Learning (RRL) provides a framework for combining a conventional feedback controller with reinforcement learning. The goal is to use the conventional controller to perform the parts of the task it can be handle, while reinforcement learning is used to address the residual part of the task. [10] This is done by superposing the output of the conventional controller to the output of the reinforcement learning policy. In our case, we use uncalibrated visual servoing as the conventional controller and TD3 as the reinforcement learning controller.

*2) Jump Start Reinforcement Learning:* Jump Start Reinforcement Learning (JSRL) is a meta algorithm for using a guide policy to accelerate the learning of a exploration policy to improve sample efficiency [11]. JSRL works by using the guide policy to generate a curriculum of starting states for the exploration policy by sampling the guide policy. Once the performance of the combined policy exceeds a threshold, the contribution of the guide policy is gradually reduced until it is no longer used. In our case, we use uncalibrated visual servoing as the guide policy and TD3 as the exploration policy.

## V. EXPERIMENTS

Our experiment involves the WAMVisualReach environment, a simulated reaching task in Mujoco [12] using a 3/4/7-DOF Barrett Whole Arm Manipulator (WAM) robot arm modified from the `FetchReach` environment [13] and `WAM Envs` [14]. The 3 DOF configuration uses the first, second, and fourth joints of the arm. The 4 DOF configuration uses the first four joints of the arm. The robot is is initialized into a start position above a table, and the goal position is randomly generated as shown in 1. The goal of the task is for the robot to move the end effector to the goal position. The state space consists of the current robot joint angles $\mathbf{q}$, imaged points tracking the end effector $(u_{e_1}, v_{e_1}, u_{e_2}, v_{e_2})$ and the goal position $(u_{g_1}, v_{g_1}, u_{g_2}, v_{g_2})$. The action space consists of an update vector of the robot's joint angles $\Delta\mathbf{q}$. We treat the task as an episodic task, where the episode ends when the end effector is within a threshold distance $\epsilon = 0.03$ of the goal position. Every 1000 environment steps, we evaluate the agent by calculating the success rate over 100 episodes. We evaluate sample efficiency by measuring the number of environment steps needed to learn a policy that has a success rate of at least 90%. We repeat each experiment 5 times with different random seeds.

The code for the `WAMVisualReach` environments are available at `https://github.com/steventango/gym-wam`.



(a) 3/4 DOF (b) 7 DOF

Fig. 1: Reaching task experimental setup, blue: end effector, red: goal position

### A. Uncalibrated Visual Servoing

We present results for uncalibrated visual servoing in Table II. We find that uncalibrated visual servoing is much more sample efficient than the reinforcement learning methods, achieving 100% success rate after the Jacobian is initialized with central differences.

### B. Reinforcement Learning

We first compare the effect of different reward functions and different representations like neural networks and Neural Jacobians on the sample efficiency and performance of reinforcement learning.

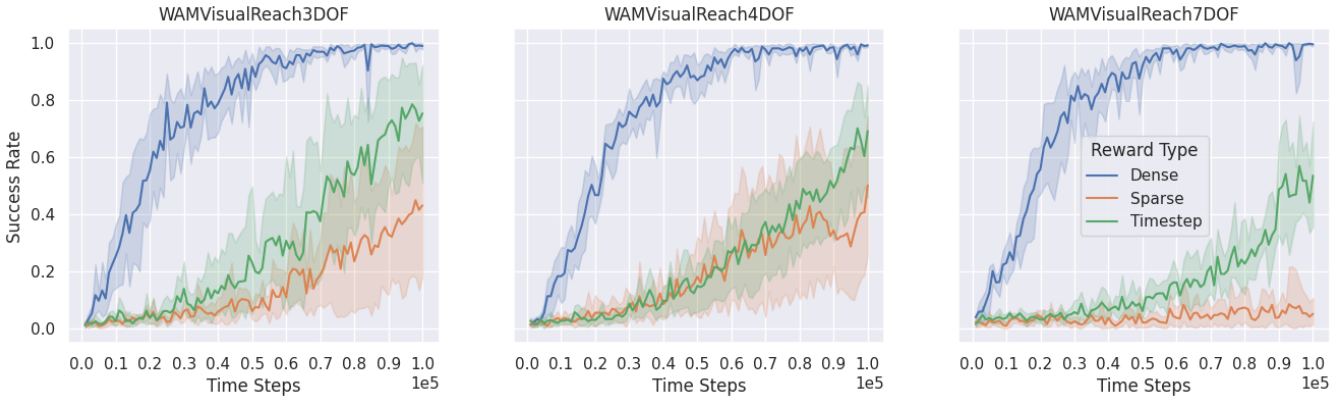



Fig. 2: Success rate of different reward functions on WAMVisualReach environment with 95% confidence intervals

*1) Reward Function:* As seen in Fig. 2, we found that the dense reward function was the most sample efficient, followed by the timestep reward function, with the sparse reward function being the least sample efficient. Notably, the 3 DOF results in Table I show that the dense reward function is at least two times as sample efficient in comparison to the other reward functions. This demonstrates that reward shaping can significantly improve sample efficiency.

TABLE I: Reward Functions on WAMVisualReach 3DOF

| Reward Function | Success Rate | Sample Efficiency |
|---|---|---|
| Sparse | $0.07 \pm 0.08$ | $> 97000$ |
| Timestep | $0.16 \pm 0.13$ | $> 84500$ |
| Dense | $\mathbf{0.93 \pm 0.13}$ | $\mathbf{41000 \pm 11367}$ |

*2) Neural Networks:* We evaluated the performance of neural network architectures with different number of layers and neurons per layer.

Observing Fig. 4, we found that overall 2 layer neural networks performed the best. We found that increasing the number of neurons per layer improved sample efficiency with diminishing returns. In Fig. 4, we note that 512 neurons per layer achieved a similar sample efficiency as 1024 neurons per layer. The remainder of experiments are conducted with 2 layer neural networks with 512 neurons per layer.

We note that smaller neural network architectures, such as 1 layer of 128 neurons (18194 parameters) are capable of achieving a final success rate of greater than 0.9, however, they are far less sample efficient.

*3) Neural Jacobian:* We find that the Neural Jacobian works well for 3 and 7 DOF robot, improving sample efficiency significantly. In fact, 12000 steps is approximately equivalent to 1600 episodes of experience, which could be collected in less than a day on a robot.

However, this technique does not work well for the 4 DOF robot. We hypothesize that this is because during training the 4 DOF robot has a tendency to move into singular configurations, which causes the Neural Jacobian to become ill-conditioned, resulting in divergence during training.

### C. Combined Approaches

In this section, we compare methods that combine uncalibrated visual servoing and reinforcement learning.

*1) Residual Reinforcement learning:* We thought that RRL would be a promising technique for combining visual servoing and reinforcement learning. However, we found that it did not work very well for our task. While RRL was able to improve the performance in the initial steps of training, it did not lead to a significant improvement in sample efficiency when compared to TD3, and even performed worse in the 4 and 7 DOF cases.

*2) Jump Start Reinforcement Learning:* We find that JSRL works well for improving the sample efficiency of the robot. A nice property of this technique is that during early training the policy still has a reasonably high success rate, and can be considered more unlikely to fail completely, which is important for training safely on a real robot. Another advantage of JSRL in contrast to Residual Reinforcement Learning is that once the target policy has been trained, it no longer has a dependency on the guide policy, so it can be used independently. We find for the reaching task, we can remove the guide policy after only 20000 steps of training.

TABLE II: WAMVisualReach Results

| Method | DOF | Success Rate | Reward | Sample Efficiency |
|---|---|---|---|---|
| UVS | 3 | $1.00 \pm 0.00$ | $-1.38 \pm 0.05$ | $60 \pm 0$ |
| | 4 | $1.00 \pm 0.00$ | $-1.35 \pm 0.08$ | $80 \pm 0$ |
| | 7 | $1.00 \pm 0.00$ | $-1.16 \pm 0.08$ | $140 \pm 0$ |
| TD3 | 3 | $0.87 \pm 0.22$ | $-1.46 \pm 0.39$ | $32400 \pm 7499$ |
| | 4 | $0.95 \pm 0.03$ | $-1.22 \pm 0.06$ | $39800 \pm 6431$ |
| | 7 | $0.96 \pm 0.03$ | $-1.14 \pm 0.11$ | $33200 \pm 2561$ |
| TD3-NJ | 3 | $1.00 \pm 0.00$ | $-1.18 \pm 0.09$ | $12200 \pm 1939$ |
| | 4 | $0.01 \pm 0.00$ | $-9.16 \pm 0.00$ | – |
| | 7 | $1.00 \pm 0.00$ | $-1.04 \pm 0.07$ | $10400 \pm 3499$ |
| RRL | 3 | $0.93 \pm 0.08$ | $-1.35 \pm 0.20$ | $> 50000$ |
| | 4 | $0.83 \pm 0.16$ | $-1.77 \pm 0.70$ | $> 50000$ |
| | 7 | $0.79 \pm 0.11$ | $-2.05 \pm 0.80$ | $> 50000$ |
| JSRL | 3 | $0.96 \pm 0.03$ | $-1.24 \pm 0.10$ | $30400 \pm 11741$ |
| | 4 | $0.98 \pm 0.01$ | $-1.12 \pm 0.06$ | $18800 \pm 2638$ |
| | 7 | $0.92 \pm 0.09$ | $-1.30 \pm 0.22$ | $21200 \pm 4956$ |


Fig. 3: Success rate on WAMVisualReach environment with different algorithms with 95% confidence intervals

### D. Qualitative Evaluation

Videos of the above experiments for qualitative evaluation are available at `https://drive.google.com/drive/folders/1xF8Z_O7cWxLBhlskcR3WSw8cPf_iActh`.

We note that the reinforcement learning based methods generally have a more direct trajectory compared to uncalibrated visual servoing. This is reflective of the fact that reinforcement learning is capable of learning a global visuomotor policy for the task.

The code for our experiments is available at `https://github.com/steventango/visual-servoing`.

## VI. CONCLUSION

We assessed the sample efficiency and performance of uncalibrated visual servoing and reinforcement learning for a reaching task. We found that techniques that combine visual servoing and reinforcement learning can greatly improved the sample efficiency of reinforcement learning.

Future work includes comparing the performance of the different methods on a real robot and comparing the different methods on a more challenging task such as a pick and place task. Further research into improving the training stability of the Neural Jacobian representation with reinforcement learning may also be promising.

## ACKNOWLEDGMENT

We thank Dylan Miller for his advice on uncalibrated visual servoing and reinforcement learning environment setup.

We acknowledge the support of the Natural Sciences and Engineering Research Council of Canada (NSERC), [funding reference number 582453].

Cette recherche a été financée par le Conseil de recherches en sciences naturelles et en génie du Canada (CRSNG), [numéro de référence 582453].

## REFERENCES

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

[2] S. Levine, C. Finn, T. Darrell, and P. Abbeel, "End-to-end training of deep visuomotor policies," *CoRR*, vol. abs/1504.00702, 2015. [Online]. Available: http://arxiv.org/abs/1504.00702

[3] F. Chaumette and S. Hutchinson, "Visual servo control. i. basic approaches," pp. 82–90, dec 2006. [Online]. Available: https://doi.org/10.1109/mra.2006.250573

[4] M. Jagersand, O. Fuentes, and R. Nelson, "Experimental evaluation of uncalibrated visual servoing for precision manipulation," in *Proceedings of International Conference on Robotics and Automation*. IEEE, 1997. [Online]. Available: https://doi.org/10.1109/robot.1997.606723

[5] A. Farahmand, A. Shademan, M. Jagersand, and C. Szepesvari, "Model-based and model-free reinforcement learning for visual servoing," in *2009 IEEE International Conference on Robotics and Automation*. IEEE, may 2009. [Online]. Available: https://doi.org/10.1109/robot.2009.5152834

[6] Y. Yuan and A. R. Mahmood, "Asynchronous reinforcement learning for real-time control of physical robots," 2022.

[7] M. Przystupa, M. Dehghan, M. Jagersand, and A. R. Mahmood, "Analyzing neural jacobian methods in applications of visual servoing and kinematic control," 2021. [Online]. Available: https://arxiv.org/abs/2106.06083

[8] S. Fujimoto, H. van Hoof, and D. Meger, "Addressing function approximation error in actor-critic methods," 2018.

[9] A. Raffin, A. Hill, A. Gleave, A. Kanervisto, M. Ernestus, and N. Dormann, "Stable-baselines3: Reliable reinforcement learning implementations," *Journal of Machine Learning Research*, vol. 22, no. 268, pp. 1–8, 2021. [Online]. Available: http://jmlr.org/papers/v22/20-1364.html

[10] T. Johannink, S. Bahl, A. Nair, J. Luo, A. Kumar, M. Loskyll, J. A. Ojea, E. Solowjow, and S. Levine, "Residual reinforcement learning for robot control," 2018. [Online]. Available: https://arxiv.org/abs/1812.03201

[11] I. Uchendu, T. Xiao, Y. Lu, B. Zhu, M. Yan, J. Simon, M. Bennice, C. Fu, C. Ma, J. Jiao, S. Levine, and K. Hausman, "Jump-start reinforcement learning," 2022. [Online]. Available: https://arxiv.org/abs/2204.02372

[12] E. Todorov, T. Erez, and Y. Tassa, "Mujoco: A physics engine for model-based control," in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2012, pp. 5026–5033.

[13] M. Plappert, M. Andrychowicz, A. Ray, B. McGrew, B. Baker, G. Powell, J. Schneider, J. Tobin, M. Chociej, P. Welinder, V. Kumar, and W. Zaremba, "Multi-goal reinforcement learning: Challenging robotics environments and request for research," 2018.

[14] K. Johnstonbaugh, "Barrett wam environments in openai gym," https://github.com/KerrickJohnstonbaugh/wam_envs, 2022.

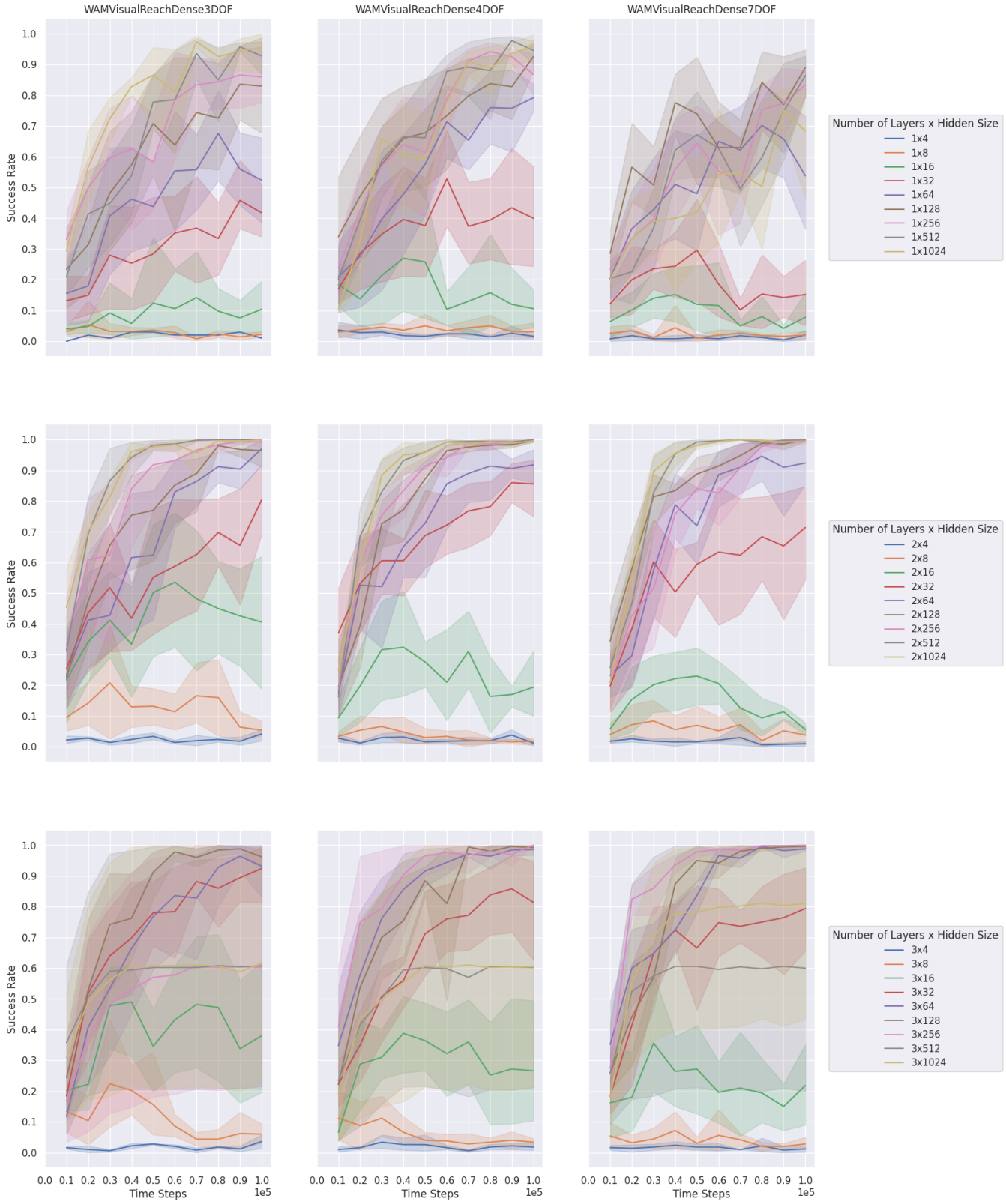



Fig. 4: Success rate on WAMVisualReach with different neural network architectures with 95% confidence intervals