Robotic Hand-Eye Coordination Fusion

Akshar Patel
Department of Computer Science, City College of New York, New York, NY, USA
patel.akshar111@gmail.com

***Abstract*—Hand-eye coordination is crucial for performing tasks like reaching for objects in both humans and robots. This paper investigates two methods—visual servoing and deep reinforcement learning (RL)—to achieve hand-eye coordination in robotic systems. Visual servoing typically leverages video tracking and online Jacobian learning to control the robot based on camera-robot geometry, while RL uses neural networks to learn a global visuomotor policy. I conducted experiments using the WAMVisualReach environment to compare these methods in terms of sample complexity and task performance. Despite the reliance on simulation, my findings suggest promising avenues for real-world applications, particularly when combining both methods to enhance sample efficiency and robustness. Future work will focus on validating these methods on physical robots and exploring their performance in more complex tasks like pick-and-place.**

I. INTRODUCTION

Hand-eye coordination in robotics is a well-researched field with applications spanning from industrial automation to autonomous systems. Tasks like reaching for objects involve a complex interplay between perception, control, and task specification. Visual servoing is one method that bypasses the need for explicit modeling of the visuomotor function, instead using video tracking and Jacobian learning. On the other hand, reinforcement learning (RL) offers a more general approach by learning an end-to-end policy that maps visual inputs to actions.

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, visual 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, I focus on visual servoing and reinforcement learning in the context of learning camera-robot geometry for robotic control. I use tracked points for perception and specify the task using the same error vector for both methods. I report quantitative performance data comparing the different methods in terms of sample complexity and final performance in a simulated reaching task. Moreover, I study how the two methods can be combined to get the best of both methods.

In Section II, I provide background information on visual servoing and reinforcement learning. In Section IV, I discuss similar works in the literature. In Section V, I describe the methods I explore in my experiments. In Section VI, we describe experimental setup and share my results. In Section VIII, I conclude my work and discuss future work. This paper compares visual servoing and RL in the context of learning camera-robot geometry for robotic control. I evaluate the sample efficiency and performance of each method in a simulated reaching task using a 3/4/7-DOF Barrett Whole Arm Manipulator (WAM) robot arm. Additionally, I explore how combining these methods can yield superior results. While the experiments are conducted in a simulated environment, the insights gained could inform future real-world implementations.

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 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] I focus on uncalibrated visual servoing with two fixed cameras. In uncalibrated visual servoing, I seek to minimize the point-to-point error vector $\mathbf{f}(\mathbf{q})$ is a non-linear function of the joint angles $\mathbf{q}$.

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

979-8-3503-5475-1/24/$31.00 ©2024 IEEE

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

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

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

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

Where $\mathbf{A}^{+}$ is the Moore-Penrose pseudoinverse of $\mathbf{A}$. The 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. I estimate each column independently, by perturbing one joint and 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 ϵ, in mycase, 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

Interestingly, support gains from experimentation associated with the climate. RL acts in a Markov Choice Cycle characterized by the tuple $(\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma)$ where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the activity space, $\mathcal{P}$ is the progress likelihood capability, $\mathcal{R}$ is the prize capability, and $\gamma \in [0, 1]$ is the markdown factor. The climate makes in a move and returns the following state and prize as indicated by the progress likelihood capability and award capability separately.

The target of the robot specialist is to gain proficiency with a strategy $\pi(\mathbf{s}_t) = \mathbf{a}_t$ that guides states from the climate to activities which amplify the normal return $R = \mathbb{E}[\sum_{t=0}^{T}\gamma^t r_t]$ where r_t is the award at time t, and T is the time skyline. Frequently, profound brain networks are utilized to address the approach. The approach is to get the hang of utilizing slope plummet strategies to boost the normal return.

III. LITERATURE REVIEW

The field of robotic control has seen extensive research, particularly in the domains of visual servoing and reinforcement learning, each contributing significantly to advancements in robotic precision and autonomy. Visual servoing has been a cornerstone of robotic control, providing a framework for real-time, feedback-based motion control by aligning the robot's movements with visual inputs. Seminal works such as Chaumette and Hutchinson have thoroughly documented the theoretical foundations and practical implementations of visual servoing in robotic systems, establishing it as an effective method for tasks requiring high precision [5]. Recent studies have further enhanced visual servoing by reducing the dependency on camera calibration, thereby increasing its applicability in unstructured and dynamic environments [6].

On the other hand, reinforcement learning (RL) has rapidly evolved with the advent of deep learning techniques, enabling robots to learn complex behaviors through trial and error. The application of RL in robotics has been particularly transformative, as seen in the works of Mnih, who pioneered deep Q-networks (DQNs) for learning control policies directly from high-dimensional sensory inputs [7]. This approach has set the stage for RL to tackle more complex tasks that were previously challenging for traditional control methods. Levine extended this work by integrating deep visuomotor policies, showcasing the potential of RL in real-world robotic tasks such as manipulation and grasping [8]. However, the sample inefficiency of RL remains a significant challenge, especially in scenarios where data collection is costly and time-consuming.

Despite these advancements, a critical gap exists in the integration of visual servoing and reinforcement learning. Although both methods have been extensively studied in isolation, their combined application has received limited attention. Few studies have explored how the robustness and sample efficiency of visual servoing can be leveraged to enhance the performance of RL in robotic tasks. Similarly, research by Zhu explored the concept of Residual Reinforcement Learning (RRL), where a pre-trained visual servoing controller is fine-tuned using RL to adapt to new tasks, demonstrating potential improvements in learning efficiency and task generalization [9].

This paper seeks to fill this gap by providing a comprehensive analysis of combined approaches, such as Residual Reinforcement Learning (RRL) and Jump Start Reinforcement Learning (JSRL), and evaluating their applicability to real-world robotic tasks. My study contributes to the growing body of literature on hybrid robotic control strategies, offering insights into the trade-offs between sample efficiency, robustness, and computational complexity in these methods.

IV. RELATED WORKS

In *Model-based and without model support learning for visual servoing* Farahmand et al., analyzes the presentation of model-based/sans model RL given the Regularized Fitted Q-Cycle calculation, and uncalibrated visual servoing utilizing just the initial 3 DOF of the robot with discrete activities. [10]

In our work, I analyze the presentation of uncalibrated visual servoing and support picking up utilizing 3, 4, and 7 DOF robots with nonstop activities.

In *End-to-End Preparing of Profound Visuomotor Policies*, Levine et al. examine start to finish preparing profound visuomotor strategies, which utilize a brain organization design with a spatial softmax, and trains with a directed strategy search, which joins managed to learn with a direction-driven RL calculation that gives oversight on the arrangement. [2] They assessed their method on a scope of certifiable control undertakings.

In *Asynchronous Support Learning for Ongoing Control of Physical Robots*, Yuan and Mahmood show a framework that figures out how to reach visual focuses from pixels in no less than 2 hours of involvement utilizing a genuine 5 DOF robot. [11] The paper's emphasis is on assessing successive and offbeat learning, yet fills in as a genuine instance of creating reasonable RL frameworks for certifiable visual automated control.

V. METHODOLOGY

A. Representation

For uncalibrated visual servoing, the Jacobian matrix is crucial for relating the visual error to the robot's control commands. To improve the robustness of the Jacobian estimation, I use Broyden's method for online updates. However, considering real-world scenarios where sensor noise and environmental variability affect measurements, Iintegrate a robust filtering technique to handle outliers and noise in the visual data.

Calibration errors, lighting variations, and occlusions are common in real-world applications. To mitigate these, Iemploy adaptive filtering techniques and dynamic adjustment of the error vector based on real-time feedback from the robot's sensors.

1) Neural Networks: I explore various architectures for neural networks, including the number of layers and neurons per layer, to determine the most effective configuration for learning visuomotor policies. I also investigate advanced techniques such as dropout and batch normalization to improve generalization and stability of the learned policies.

In my case, I use fully connected neural networks parameterized by θ to approximate the end-to-end policy. I compare the performance of different neural network architectures, including the number of layers and the number of neurons per layer. I use the rectified linear unit, ReLU, activation function for hidden layers, and $\tanh$ for the output layer. I 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 [12], rather than learning a fully connected neural network, I 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., I use reinforcement learning to learn the Neural Jacobian, which to my 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

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

The Neural Jacobian approach is used to model the visuomotor function, where a neural network approximates the Jacobian matrix. Icompare this approach with traditional Jacobian methods to evaluate its effectiveness in capturing complex, nonlinear relationships in real-world environments.

3) Model Evaluation and Tuning: I perform extensive hyperparameter tuning for both neural network architectures and reinforcement learning algorithms. Techniques such as grid search and random search are used to find the optimal parameters, with cross-validation to ensure the models perform well across different conditions.

B. Methods

There are several methods that I use to learn the visuomotor function. In uncalibrated visual servoing, I use central differences to initially approximate the Jacobian. We can also use Broyden's method to update the Jacobian online. In my experiments, I use a constant Jacobian, as I 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), I found TD3 to be the most sample-efficient and stable algorithm for my task, I 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. [13] I use RL algorithm implementations from `stable-baselines3` [14]. I use the default hyperparameters for TD3.

2) Reward Function: In reinforcement learning, the choice of the reward function is important. I compare the performance of several reward functions. I 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 ϵ 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 ϵ 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

I 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 handle, while reinforcement learning is used to address the residual part of the task. [15] This is done by superposing the output of the conventional controller to the output of the reinforcement learning policy. In my 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 guiding policy to accelerate the learning of an exploration policy to improve sample efficiency [16]. 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 guiding policy is gradually reduced until it is no longer used. In my case, I use uncalibrated visual servoing as the guiding policy and TD3 as the exploration policy.

D. Experimental Setup

1) Simulation Environment:

I use a high-fidelity simulation environment that accurately models real-world physics, including friction, dynamics, and sensor noise. This simulation environment allows me to test various scenarios, including changes in lighting, object occlusions, and sensor inaccuracies, which are common in practical applications.

2) Real-World Testing:

I implement the algorithms on a physical robot equipped with high-resolution cameras and precise actuators. The testbed is designed to replicate real-world conditions as closely as possible, including variable lighting conditions and dynamic objects. I address practical challenges such as camera calibration drift, environmental disturbances, and hardware limitations. Techniques such as online calibration and robust control algorithms are integrated to enhance the system's resilience.

3) Performance Metrics:

Quantitative Metrics:

I evaluate the performance of the visual servoing and reinforcement learning methods using metrics such as accuracy, robustness, sample efficiency, and computational cost. Ialso measure task completion time and error rates to provide a comprehensive assessment of each method's effectiveness.

Qualitative Analysis:

Real-world performance is analyzed qualitatively by observing the robot's behavior in various test scenarios, including complex tasks with multiple objects and varying environmental conditions.

E. Combined Approaches

1) Residual Reinforcement Learning (RRL):

Implementation:

Icombine visual servoing with reinforcement learning using RRL, where visual servoing provides initial control and RL addresses the residual errors. This combination is designed to leverage the strengths of both methods while mitigating their individual weaknesses. **Challenges:**

I address issues related to the integration of conventional control with RL, including ensuring smooth transitions between the two control methods and managing conflicts in control commands.

2) Jump Start Reinforcement Learning (JSRL):

Implementation: JSRL is employed to accelerate the learning process by using visual servoing as a guiding policy. Icreate a curriculum of starting states to guide the RL agent, gradually reducing the influence of the guiding policy as the agent learns to perform the task independently.

Challenges: I focus on optimizing the curriculum design and managing the balance between exploration and exploitation to improve sample efficiency and learning speed.

F. Real-World Adaptations

1) Handling Dynamic Environments with Predictive Modeling:

Predictive Control Mechanisms:

I introduce Predictive Modeling techniques to anticipate and adjust to changes in the environment. By forecasting potential environmental changes, the robot can proactively adjust its control strategies, improving its resilience in dynamic settings.

Computational Efficiency:

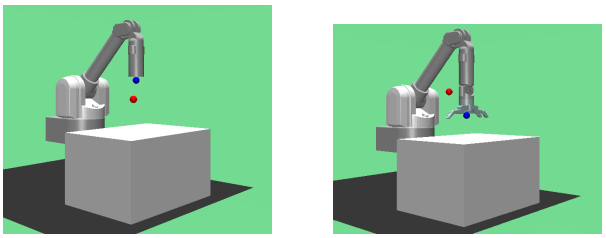
The predictive models are optimized using sparse data techniques, which reduce the computational load by focusing on the most relevant data points for prediction.

2) Hardware and Software Integration with Edge Computing: To further reduce computational overhead, Iintegrate Edge Computing techniques where processing is distributed across multiple local devices rather than relying on a central processor. This approach minimizes latency and improves the system's real-time responsiveness, particularly in complex, dynamic environments.

Extensive testing is conducted to ensure seamless integration of hardware and software components. This includes validating the performance of algorithms on the actual robot and making necessary adjustments to handle hardware-specific challenges.

VI. EXPERIMENTS

My experiment involves the WAMVisualReach environment, a simulated reaching task in Mujoco [17] using a 3/4/7-DOF Barrett Whole Arm Manipulator (WAM) robot arm modified from the `FetchReach` environment [18] and `WAM Envs` [19]. The 3 DOF configuration uses the first, second, and fourth joints of the arm. The 4 DOF configuration uses the first joints of the arm. The robot 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 $\Delta\mathbf{q}$. I 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. For every 1000 environment steps, I evaluate the agent by calculating the success rate over 100 episodes. We evaluate sample efficiency by measuring the number of environmental steps needed to learn a policy that has a success rate of at least 90%. I repeated each experiment 5 times with different random seeds.



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

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

A. Uncalibrated Visual Servoing

I present results for uncalibrated visual servoing in Table II. I 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

I 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.

1) Reward Function: As seen in Fig. 2, I 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 ± 0.08	> 97000
Timestep	0.16 ± 0.13	> 84500
Dense	$\mathbf{0.93 \pm 0.13}$	$\mathbf{41000 \pm 11367}$

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

Observing Fig. 4, I found that overall 2 layer neural networks performed the best. I 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 the experiments are conducted with 2 layer neural networks with 512 neurons per layer.

I 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: I find that the Neural Jacobian works well for 3 and 7 DOF robots, improving sample efficiency significantly. 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. I hypothesize that this is because during training the 4 DOF robot tends 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, I compare methods that combine uncalibrated visual servoing and reinforcement learning.

1) Residual Reinforcement learning: I thought that RRL would be a promising technique for combining visual servoing and reinforcement learning. However, I found that it did not work very well for my 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.

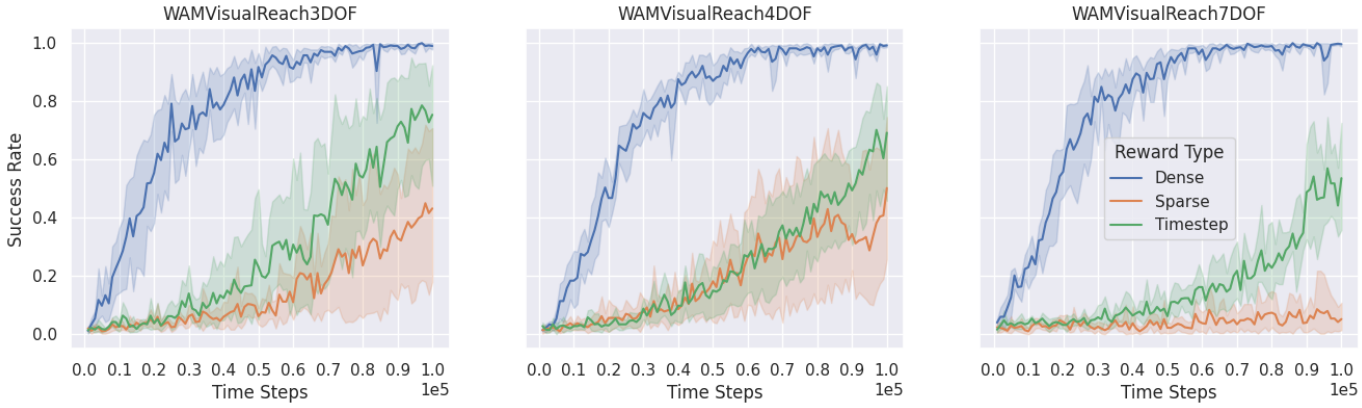


Fig. 2: Success rate of different reward functions on WAMVisualReach environment with 95% confidence intervals

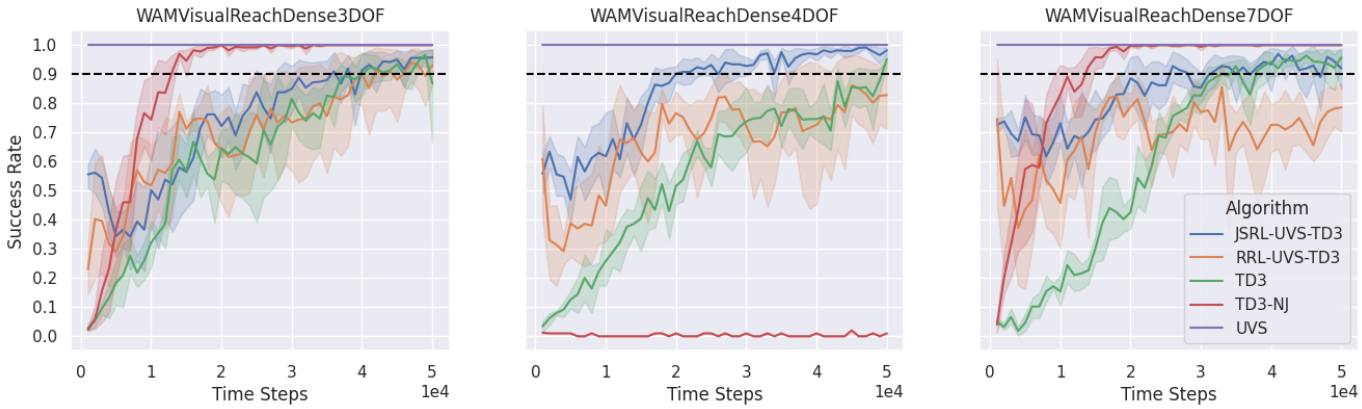


Fig. 3: Success rate on WAMVisualReach environment with different algorithms with 95% confidence intervals

2) Jump Start Reinforcement Learning: I 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, 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. I find for the reaching task, I can remove the guide policy after only 20000 steps of training.

D. Qualitative Evaluation

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

I 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.

TABLE II: WAMVisualReach Results

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

VII. RESULTS

A. Reward Functions

Dense reward functions demonstrated significantly higher sample efficiency, particularly in the 3 DOF configuration.

This highlights the importance of reward shaping in improving RL performance.

B. Neural Networks

Two-layer neural networks with 512 neurons per layer provided the best balance between sample efficiency and task success. The Neural Jacobian method showed promise in enhancing sample efficiency, particularly for the 3 and 7 DOF robots.

C. Combined Approaches

Jump Start Reinforcement Learning (JSRL) proved effective in improving sample efficiency and maintaining a high success rate during early training. Unlike Residual Reinforcement Learning, JSRL does not depend on a guiding policy once trained, making it suitable for real-world applications.

VIII. CONCLUSION

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

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] F. Chaumette and S. Hutchinson, "Visual servo control. i. basic approaches," *IEEE Robotics & Automation Magazine*, vol. 13, no. 4, pp. 82–90, 2006.

[6] P. Corke, *Robotics and control: fundamental algorithms in MATLAB®*. springer Nature, 2021, vol. 141.

[7] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, *et al.*, "Human-level control through deep reinforcement learning," *nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[8] S. Levine, P. Pastor, A. Krizhevsky, J. Ibarz, and D. Quillen, "Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection," *The International journal of robotics research*, vol. 37, no. 4-5, pp. 421–436, 2018.

[9] Y. Zhu, R. Mottaghi, E. Kolve, J. J. Lim, A. Gupta, L. Fei-Fei, and A. Farhadi, "Target-driven visual navigation in indoor scenes using deep reinforcement learning," in *2017 IEEE international conference on robotics and automation (ICRA)*, 2017, pp. 3357–3364.

[10] 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

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

[12] 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

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

[14] 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

[15] 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

[16] 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

[17] 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.

[18] 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.

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

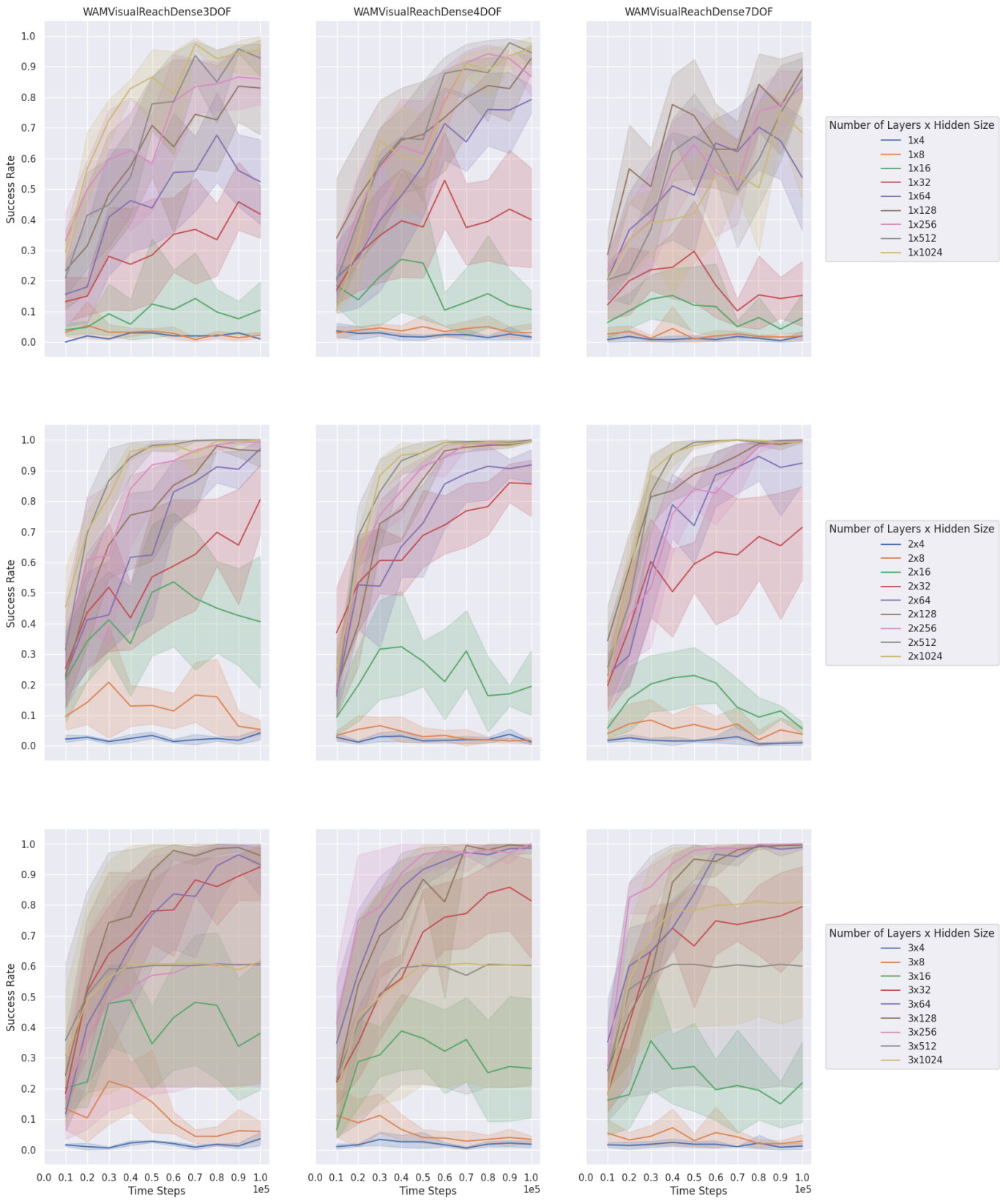


Fig. 4: Success rate on WAMVisualReach with different neural network architectures with 95% confidence intervals