IEEE Robotics and Automation Letters (RAL) paper presented at
2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
October 14-18, 2024. Abu Dhabi, UAE

RESPRECT: Speeding-up Multi-fingered Grasping with Residual Reinforcement Learning

Federico Ceola, Lorenzo Rosasco and Lorenzo Natale

***Abstract*—Deep Reinforcement Learning (DRL) has proven effective in learning control policies using robotic grippers, but much less practical for solving the problem of grasping with dexterous hands – especially on real robotic platforms – due to the high dimensionality of the problem.**

In this work, we focus on the multi-fingered grasping task with the anthropomorphic hand of the iCub humanoid. We propose the *RESidual learning with PREtrained CriTics* (RESPRECT) method that, starting from a policy pre-trained on a large set of objects, can learn a residual policy to grasp a novel object in a fraction ($\sim 5\times$ faster) of the timesteps required to train a policy from scratch, without requiring any task demonstration. To our knowledge, this is the first Residual Reinforcement Learning (RRL) approach that learns a residual policy on top of another policy pre-trained with DRL. We exploit some components of the pre-trained policy during residual learning that further speed-up the training. We benchmark our results in the iCub simulated environment, and we show that RESPRECT can be effectively used to learn a multi-fingered grasping policy on the real iCub robot.

The code to reproduce the experiments is released together with the paper with an open source license[1].

***Index Terms*—Dexterous Manipulation; Multifingered Hands; Reinforcement Learning.**

I. Introduction

LEARNING dexterous manipulation tasks is an open challenge in robotics [1]. These tasks require controlling tens of degrees of freedom (DoFs), to deal with possibly imprecise perception of the environment, and to manage hand-object interactions. Grasping is the key task to enable the execution of other dexterous manipulation tasks such as object re-orientation [1], [2].

Manuscript received: September 6, 2023; revised: December 13, 2023; accepted: January 19, 2024.

This paper was recommended for publication by Editor Aleksandra Faust upon evaluation of the Associate Editor and Reviewers' comments. This work was supported by the PNRR MUR project PE0000013-FAIR. L. R. acknowledges the financial support of the European Research Council (grant SLING 819789), the European Commission (ELIAS 101120237), the US Air Force Office of Scientific Research (FA9550-18-1-7009 and FA8655-22-1-7034), the Ministry of Education, University and Research (grant ML4IP R205T7J2KP) and the Center for Brains, Minds and Machines (CBMM), funded by NSF STC award CCF-1231216. This work represents only the view of the authors. The European Commission and the other organizations are not responsible for any use that may be made of the information it contains.

Federico Ceola and Lorenzo Natale are with Humanoid Sensing and Perception (HSP), Istituto Italiano di Tecnologia (IIT), Genoa, Italy (email: `name.surname@iit.it`).

Federico Ceola and Lorenzo Rosasco are with Laboratory for Computational and Statistical Learning (LCSL), with Machine Learning Genoa Center (MaLGa) and with Dipartimento di Informatica, Bioingegneria, Robotica e Ingegneria dei Sistemi (DIBRIS), University of Genoa, Genoa, Italy.

Lorenzo Rosasco is also with Istituto Italiano di Tecnologia (IIT) and with Center for Brains, Minds and Machines (CBMM), Massachusetts Institute of Technology (MIT), Cambridge, MA (email: `lrosasco@mit.edu`).

[1]https://github.com/hsp-iit/rl-icub-dexterous-manipulation

The latest model-free DRL approaches, such as SAC [3] or PPO [4], are feasible options to learn problems with high-dimensionality. However, their deployment on real robotic platforms is difficult due to the huge amount of timesteps required to explore the environment at the beginning of the training. A major trend in the literature to overcome this problem is to train a policy in simulation and transfer it on the real robot [5]. However, the success of these approaches depends on the sim-to-real gap between the simulated and the real environments. With our work, we aim at overcoming this limitation by proposing a method for fast learning of the multi-fingered grasping task.

The contribution of this paper is a method that we call RESPRECT. The core of this algorithm is an RRL method that leverages on a pre-trained base policy to learn a residual additive policy on a novel object. In contrast to existing residual learning methods, which usually rely on classical closed-loop controllers as base policies, we show that it is possible to learn a residual policy also if a hand-tuned closed-loop base controller is unavailable, as for the considered multi-fingered grasping task.

As a further contribution, we propose to exploit pre-trained components to speed-up the residual training. We initialize the SAC *Critics* components in the residual policy using the weights of the pre-trained policy. In the experimental section we show that this leads to a significant reduction of the training time, making it suitable for learning policies directly in the real world.

We benchmark our results against conventional fine-tuning and Meta Reinforcement Learning (MetaRL) approaches in a simulated environment with the 9-DoFs hand of the iCub humanoid [6], surpassing all the considered baselines in most of the experiments. Furthermore, the experimental results show that RESPRECT achieves the same success rate as the state-of-the-art method for multi-fingered grasping G-PAYN [7], while requiring only a fraction of the training timesteps ($1M$ compared to $5M$). Notably, differently from G-PAYN [7], RESPRECT accomplishes this without the need for grasping demonstrations to initialize the training process. While these requirements prevent G-PAYN [7] from learning grasping policies on the real robot, we deploy RESPRECT on the real iCub, showing that it can learn a grasping policy for two new objects in ~ 2.5 hours and ~ 30 minutes.

To the best of our knowledge, this is the first DRL algorithm that has been successfully used to solve the problem of grasping with an articulated hand with several DoFs directly on the real robot from visual, tactile and proprioceptive data, and without the need for task demonstrations.

II. Related Work

Multi-fingered Grasping The literature mainly focuses on grasp poses generation [8] or detection of finger-object contact points [9], [10], whereas the deployment of efficient control strategies to perform the grasp has received less attention. Recent work proposes to solve the problem with DRL-based approaches. The work in [11] starts from a grasp pose given by an external algorithm, and relies on synergies to reduce the number of DoFs to control the fingers of a Shadow hand. The policy is trained on a multi-modal input comprising tactile information, joint angles and torques, which are not always available in other robotic hands. Moreover, it does not take into account information about the object (e.g., visual feedback of the environment or object pose) during grasp execution to allow grasp recovery if the initial grasp pose is unfeasible. G-PAYN [7], instead, considers as proprioceptive information the cartesian pose of the end-effector and finger joint positions, relying on visual feedback from the head-mounted camera of the robot. However, training is performed in simulation with long training sessions relying on huge amounts of grasping demonstrations, and therefore it cannot be performed on the real robot. DexPoint [5], instead, learns to grasp and open a door from pointclouds, showing that sim-to-real transfer can be obtained without fine-tuning. While being an interesting research direction, this transfer is strongly affected by sensors quality. This is particularly evident with objects in proximity to depth sensors. We overcome these limitations providing a method that can be deployed on a real robot for fast learning of multi-fingered grasping from visual, tactile, and proprioceptive data. Notably, differently from the state-of-the-art G-PAYN [7], RESPRECT does not require any task demonstration, while being trained much faster.

Fast DRL State-of-the-art DRL algorithms, e.g. SAC [3] or PPO [4], are sample-inefficient and require huge amounts of training episodes to be optimized, hindering their application on tasks that require to be trained on real robots. Some methods adapt standard DRL algorithms leveraging off-line task demonstrations to overcome this limitation. [12], [13] adapt DDPG [14] to exploit task demonstrations: DDPGfD [12] leverages on a prioritized replay mechanism to sample transitions between demonstrations and agent data, while [13] adds a Behavior Cloning (BC) loss component to the one of DDPG [14] to mimic demonstrations. DAPG [15], instead, fine-tunes a DRL policy pre-trained on demonstrations with imitation learning. Off-line DRL approaches as AWAC [16], instead, pre-train a policy with DRL on demonstrations and then adapt it on-line on the robot. Results in [7] show that these approaches are not suited for multi-fingered grasping from visual, tactile, and proprioceptive data. A different approach to adapt a DRL policy is fine-tuning. [17] compares fine-tuning to several MetaRL approaches. While being promising for fast adaptation of DRL tasks, MetaRL algorithms are often difficult to use in practice due to their complexity. For example, PEARL [18] requires learning an additional network for task context inference, which is used to condition the trained policy. In some cases, MetaRL methods can be employed only with on-policy DRL algorithms [19]. We benchmark our method against fine-tuning and MetaRL baselines.

RRL RRL methods aim at speeding-up policy learning. They are designed to predict a residual action, that is added to the output of an existing controller. The first RRL approach was introduced in [20] to improve imperfect controllers in simulated manipulation tasks, such as object pushing or pick-and-place. [21], [22] propose RRL methods for insertion tasks on real robots, either from observable and measurable states, or from raw pixels. [23] proposes to modify also the signal to a base feedback controller to avoid the feedback distribution shift caused by the residual policy, which the base controller tries to resist. These approaches share a common limitation, in that they rely on manually designed conventional controllers, which are difficult to design for a multi-fingered grasping task. [24] overcomes this limitation learning a residual policy to solve insertion tasks on top of Dynamic Movement Primitive (DMP) base policies extracted through BC. However, learning DMPs from visual, tactile and proprioceptive data is impractical. [25] proposes an RRL method to improve a policy trained with BC by superimposing a residual policy trained with DRL on seven simulated manipulation tasks (the same tasks used for BC training). This removes the dependency on hand-engineered base controllers, but requires task demonstrations which are difficult to obtain on the real robot. This challenge becomes even more pronounced in the context of this work, where we aim at learning multi-fingered grasping policies on objects unseen during base policy training. Furthermore, evidence from previous work [7] shows that training policies on the task at hand with BC is impractical. Residual learning has also found application on different robotic tasks, such as object throwing [26], where the residual throwing velocity is regressed and superimposed on the velocity predicted by an ideal physics controller, or to learn navigation strategies [27], where a classical controller serves as the base policy.

We overcome the limitations of the state-of-the-art, which either depend on classical base controllers (unavailable for the considered multi-fingered grasping task), or rely on a policy trained with BC on the same task. The goal of the proposed algorithm is to allow the robot to quickly learn to grasp unseen objects, starting from a DRL policy that is pre-trained on a set of generic objects. To achieve this, we use visual features obtained from a backbone pre-trained on a dataset including egocentric images from everyday tasks, without making any assumption on the target object to be grasped. We leverage the pre-trained DRL policy even further: the residual *Critics* component is initialized using the pre-trained weights of the base policy, significantly speeding-up the training. These enhancements are inherently unattainable using the base policies employed in existing literature.

III. Methodology

A. Grasping Pipeline

We rely on the grasping pipeline introduced in [7], and consider the right hand of the iCub humanoid robot. This is actuated by 9 motors and is equipped with tactile sensors on the fingertips. The pipeline includes two main phases. In the initial

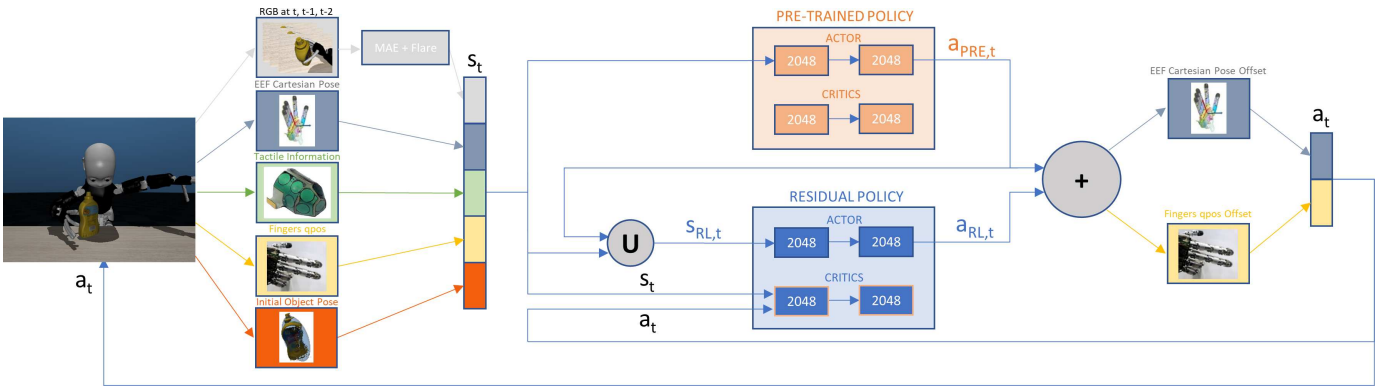

Fig. 1. RESPRECT overview. We compute state s_t from RGB images at timesteps t, $t-1$ and $t-2$ (processed through the MAE in [30] and combined with Flare [29]), end-effector cartesian pose, tactile information and finger joint poses. We compute action a_t (composed of cartesian offsets for the end-effector and finger joint offsets) combining the outputs $a_{PRE,t}$ of the pre-trained policy and $a_{RL,t}$ of the residual policy. Note that $a_{RL,t}$ is the output of the residual policy *Actor*, given the concatenation of s_t and $a_{PRE,t}$ into $s_{RL,t}$. We train only the two 2048-dimensional fully connected layers in the residual *Actor* and *Critics*. For the latter, we start from the *Critics* weigths of the pre-trained policy (orange outline). For the sake of clarity, we do not report the input of the *Critics* in the pre-trained policy, and the output of both the *Critics* being the same as the ones in SAC [3].

phase, the end-effector is moved in a pose spaced $5cm$ from a grasp pose generated by an object agnostic algorithm. This is either an algorithm based on superquadrics [8], hereinafter referred to as *Superquadrics*, or VGN [28]. Subsequently, we employ a DRL policy, trained with the proposed RESPRECT, to approach and lift the object. The DRL policy controls the 6 DoFs of the end-effector's pose and the 9 finger joints to approach the object and finally lift it.

We consider five elements as the state of the Markov Decision Process (MDP) underlying the DRL policy at hand. We use the visual Flare [29] features extracted with the *ViT-Large* model from the Masked Autoencoder presented in [30] (MAE). This model was trained on several real-world visual datasets, comprising Ego4D [31] that well represents the type of visual feedback acquired by the robot head-mounted camera. We also consider a binary tactile value for each fingertip, the cartesian pose of the end-effector, finger joint positions and an estimate of the initial pose of the object to grasp (the latter computed as the center of the superquadric, or the center of the segmented pointcloud, when considering VGN for grasp pose synthesis). The DRL policy outputs a 15-dimensional vector that represents offsets for moving the end-effector and finger joints. The policy is trained with a reward function that considers: the pose of the hand with respect to the estimated initial position of the object, the number of fingers in contact with the object (detected in the case of tips-object meshes contact in simulation, or when tactile sensors on the real robot are triggered), the position of the object when lifted, and the terminal condition of the episode. An episode is regarded as successfully terminated if the object is grasped and uplifted by $10cm$. Conversely, an episode is deemed a failure if the object is moved too far from the initial position, the inverse kinematics (IK) solver cannot find a solution for the specified configuration, or the number of timesteps exceeds the designated maximum threshold. For further details, we refer the reader to the description in [7].

B. Residual Policy Training

To train the grasping policy we propose a novel RRL method. This leverages on two components: a policy that is pre-trained with G-PAYN [7] in simulation, and a residual, object-specific, policy trained with a modified version of SAC [3]. The former is trained for two million timesteps using the MuJoCo models of the *Scanned Objects Dataset* (MSO) [32], [33]. In the simulated experiments, the latter is trained on seven YCB-Video [34] objects, while in the real world experiment it is trained on the *006_mustard_bottle* and on the *021_bleach_cleanser* (also taken from YCB-Video). Given the state of the system s_t at time t, our approach outputs an action a_t for the robot. This is the sum of two components: the action $a_{PRE,t}$ predicted by the pre-trained policy, and the action $a_{RL,t}$ predicted by the residual policy. While training the residual policy, the weights of the pre-trained policy remain fixed, and this is used only to predict $a_{PRE,t}$ which is needed for the optimization of the residual policy. As shown in Fig. 1, we modify the standard SAC [3] inputs for the *Actor* and *Critics* components in the residual policy. Specifically, for the *Actor*, we compute the state $s_{RL,t}$ as the concatenation between the state s_t and the action produced by the pre-trained policy $a_{PRE,t}$. This allows the policy to compute the residual action $a_{RL,t}$ conditioned not only to the current state of the system, but also to the action produced by the pre-trained component. The *Critics* are fed with the state s_t and the action a_t, instead of $a_{RL,t}$ and $s_{RL,t}$ as in the standard SAC [3] algorithm. This allows training the residual *Critics* starting with the weights of the pre-trained counterparts and to speed-up the initial stage of the training as demonstrated by the experiments.

IV. Experimental Setup

We validate our approach in the MuJoCo [35] simulated environment customized for the iCub robot in [7]. We train all the policies by adapting the SAC [3] implementation in the Stable-Baselines3 [36] library. For RESPRECT, we consider training hyperparameters as the ones in [7], but we increase

Parameter	Value
Optimizer	Adam
Learning Rate	$3 \cdot 10^{-4}$
Discount (γ)	0.99
Replay Buffer size	10^6
Number of Hidden Layers (all networks)	2
Number of Hidden Units per Layer	1024
Number of Samples per Minibatch	256
Entropy Target	-15
Non-linearity	ReLU
Target Smoothing Coefficient (τ)	0.005
Target Update Interval	1
Gradient Steps	10
Training Frequency	10 Timesteps
Total Environment Timesteps	$1 \cdot 10^6$
Entropy Coefficient	0.01

TABLE I
RESPRECT TRAINING HYPERPARAMETERS.

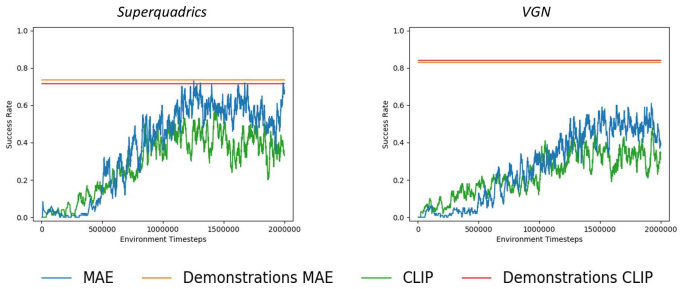

Fig. 2. We compare the success rate obtained with different visual backbones (MAE and CLIP) when learning with G-PAYN [7] on the MSO dataset for $2M$ timesteps. We report in separate plots the cases in which we use *Superquadrics* and VGN for the initial grasp pose synthesis. We also report the average success rate of the *Demonstrations* used to initialize the G-PAYN replay buffer. Note that they slightly differ for CLIP and MAE due to the random initialization of each episode to collect demonstrations.

the number of gradient steps to 10 and we set the initial entropy coefficient for SAC [3] to 0.01. We provide a complete overview of the training hyperparameters in Tab. I.

Both for RESPRECT and the baselines (see Sec. V-A), we improve the visual feature extractor with respect to [7] by replacing the pre-trained *ViT-B/32* CLIP [37] model with the pre-trained *ViT-Large* model of the MAE in [30] and we increase the number of hidden units in the 2 layers of the SAC [3] *Actor* and *Critics* to 2048. The reason for this change is that the *ViT-Large* model of the MAE led to higher success rate of the pre-trained policy. In Fig. 2 we compare G-PAYN trained on MSO with the two different feature extractors.

A. Real Robot Setup

We deploy our method on the real iCub[2] [6] humanoid. The robot is equipped with an *Intel(R) RealSense D405* on a headset for the acquisition of RGB images and depth information (the latter is not used by the DRL algorithm but only during the approach phase to compute the initial grasp pose). We rely on the YARP [38] middleware for the implementation and the communication between the different modules. For policy training, we adapt some components of the simulated training pipeline:

- We use as input RGB image to the MAE the central crop of size 320×240 of the image acquired by the camera of the robot to match the real and simulated visual fields of view.
- We adapt some components of the reward function and some of the terminal conditions, since the precise position of the target object is difficult to obtain on the real robot. Specifically, to reward the position of the object along the axis perpendicular to the table (r_{object_height} in [7]), we consider the z-component of the position of the end-effector of the robot, once it has touched the object with at least two fingers. While this is sufficient to evaluate whether an episode ends positively (i.e. the object has been grasped), it does not allow to determine failure cases when the object falls off the table or moves away from its initial condition. We overcome this problem by manually sending a notification to the learning module.
- We assume that a finger is in contact with the object when the tactile sensors mounted on the fingertip is triggered. If a fingertip is in contact with other parts of the environment (e.g. the table or other fingers in possibly dangerous configurations) we consider this as a possibly unsafe state for the robot, and we manually terminate the execution of the grasping episode.
- We move the end-effector of the robot both to initialize the grasping (i.e. in a pose spaced $5cm$ from the one estimated by the *Superquadrics*) and during policy execution via a cartesian controller that performs IK and trajectory computation [39]. We set the fixation point of the gaze of the robot to the center of the object, which is randomly placed on the table in a graspable configuration at the beginning of the grasping episode, as it is estimated by the *Superquadrics*.

We refer the reader to the code released together with the paper for the implementation details of RESPRECT on the real iCub.

V. RESULTS

To evaluate the effectiveness of our approach, we benchmark our results in the scenarios proposed in [7], using seven objects from the YCB-Video dataset chosen to represent various grasp types (the *004_sugar_box*, the *005_tomato_soup_can*, the *006_mustard_bottle*, the *008_pudding_box*, the *010_potted_meat_can*, the *021_bleach_cleanser*, and the *035_power_drill*). As in [7], for each object, we employ two different methods for the computation of the initial grasp pose: *Superquadrics* and *VGN*. We then evaluate our approach training a policy to grasp a *006_mustard_bottle* and a *021_bleach_cleanser* on the real iCub robot.

A. Baselines

We benchmark **RESPRECT** (blue line in Fig. 3), comparing the success rate for increasing environment timesteps against the following:

[2]We run the module for training the grasping policy on a machine equipped with an Intel(R) Core(TM) i7-9750H CPU @ 2.60GHz, and a single NVIDIA RTX 2080 Ti.

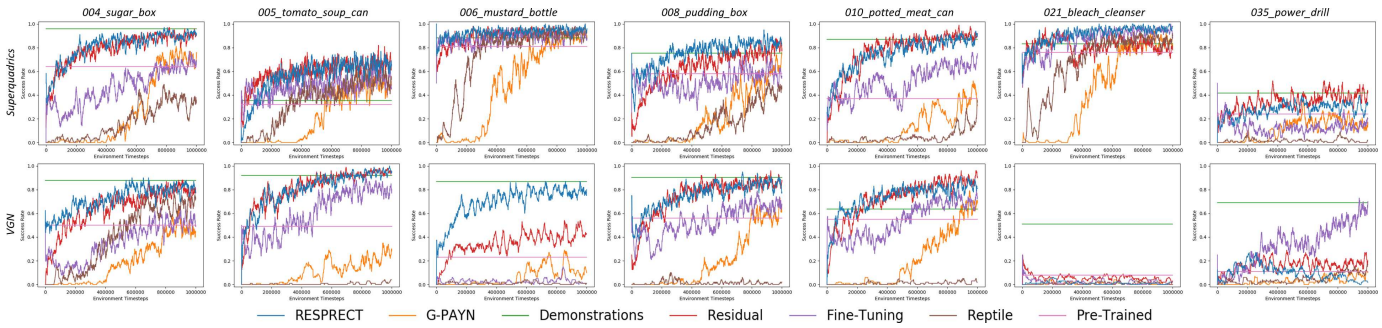

Fig. 3. **Results**. We compare the success rate achieved by **RESPRECT** to the baselines for $1M$ environment timesteps. We benchmark the performance over seven YCB-Video objects (on different columns) starting from grasp poses generated either by *Superquadrics* or *VGN* (on different rows).

- **G-PAYN** (orange line in Fig. 3): this is the approach proposed in [7]. For a fair comparison, we train from scratch **G-PAYN** on the considered objects using the pre-trained MAE [30] as feature extractor, and we increase the dimension of the two fully connected layers in the SAC *Actor* and *Critics* networks to 2048.
- **Demonstrations** (green line in Fig. 3): this is the pipeline proposed in [7] that is used to collect the demonstrations for the training of **G-PAYN**. We report the success rate of the demonstrations collected to fill the initial replay buffer in the **G-PAYN** experiment.
- **Residual** (red line in Fig. 3): this is a similar approach to **RESPRECT**, but we do not initialize the SAC *Critics* of the residual policy with the weights of the SAC instance pre-trained on MSO. We provide an overview of this approach in App. D. Note that **Residual** is a contribution itself over pre-existing methods that rely on classical base controllers, which, to the best of our knowledge are unavailable for the considered multi-fingered grasping task.
- **Fine-Tuning** (violet line in Fig. 3): we start from the policy pre-trained with **G-PAYN** on the MSO dataset and we fine-tune the fully connected layers in the *Actor* and *Critics* on the considered YCB-Video object. Differently from **RESPRECT** and **Residual**, we perform one gradient step at each training iteration. In App. A, we compare the obtained success rate to the one achieved when performing 10 gradient steps.
- **Reptile** (brown line in Fig. 3): this is an adaptation for the SAC [3] algorithm of the meta learning approach proposed in [40]. We modify the original *Reptile* algorithm as in [17], filling the initial replay buffers for the pre-training on MSO as in **G-PAYN**. We chose this method because, according to the results shown in [17], it is the most competitive among the considered MetaRL baselines on the robotic benchmark RLBench [41]. As for **Fine-Tuning**, we perform one gradient step at each training iteration (see App. B).
- **Pre-Trained** (pink line in Fig. 3): we compute the success rate achieved by the policy pre-trained on MSO over 100 randomly initialized episodes.

B. Simulation Results

Results in Fig. 3 show that **RESPRECT** and **Residual** constantly outperform the success rate obtained with **Pre-Trained**. This demonstrates the effectiveness of the proposed RRL approach.

RESPRECT manages to achieve in one million environment timesteps a comparable success rate as **G-PAYN** when the latter is trained for five million timesteps (we refer the reader to the results in App. C for a comparison with the full training of **G-PAYN**), and outperforms it when the training is stopped after one million timesteps. We also show that our approach manages to achieve a comparable success rate as the **Demonstrations** baseline in twelve experiments out of fourteen, outperforming it in seven task instances.

Compared to fine-tuning and MetaRL based approaches for fast task adaptation of a pre-trained policy, overall, **RESPRECT** performs much better than **Fine-Tuning** and **Reptile**. The only exception is represented by the *035_power_drill+VGN* experiment, where **Fine-Tuning** outperforms all the other considered methods, and the full training of **G-PAYN**. In those cases in which the baselines achieve a similar success rate to **RESPRECT**, they require a larger number of timesteps, see for instance the *010_potted_meat_can+VGN* and the *004_sugar_box+VGN* experiments.

In most of the experiments, for example the *004_sugar_box+VGN*, the success rate of **RESPRECT** in the initial training timesteps has a steeper slope than **Residual**, which is crucial to speed-up the training procedure. Moreover, we noticed that **Residual** tends to have higher *Critics* losses, which may lead to training instability. This occurs, for example in the *021_bleach_cleanser+Superquadrics* experiment, where there is a performance drop after $\sim 300k$ timesteps.

In Fig. 4, we show a qualitative evaluation of RESPRECT in the iCub simulated environment. We report an exemplar sequence in which the residual policy successfully grasps the target object, while the pre-trained policy fails starting from the same object configuration.

C. Real Robot Results

We train RESPRECT to grasp the *006_mustard_bottle* and the *021_bleach_cleanser* starting from grasp poses given by

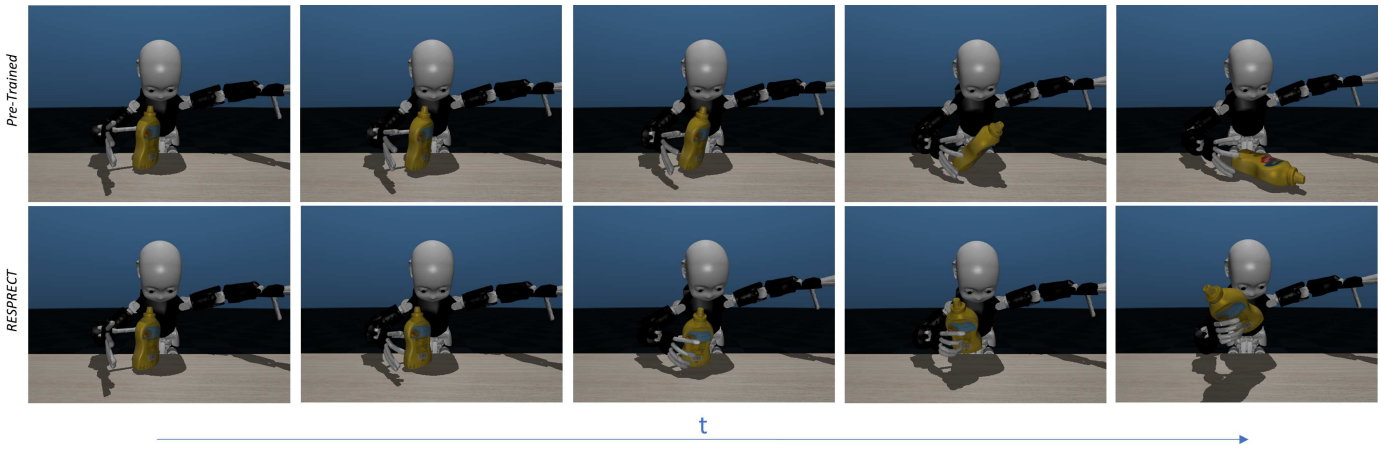

Fig. 4. Qualitative evaluation of the proposed RESPRECT. We compare it to the pre-trained policy in the same experiment. We show how the residual output of RESPRECT allows to solve the task.

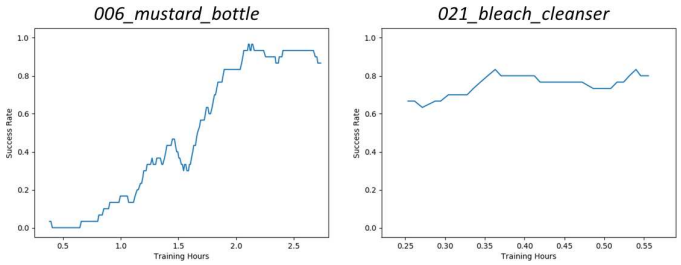

Fig. 5. RESPRECT success rate (averaged over the last 30 training episodes) for increasing training time on the real iCub robot.

Superquadrics on the real iCub humanoid. For these experiments, we consider as base policy the same policy pre-trained on the simulated MSO dataset used for the experiments in simulation. In Fig. 5, we report the success rate for increasing robot training time. Results show that after ~ 2.5 hours ($\sim 10k$ timesteps) and ~ 30 minutes ($\sim 1.5k$ timesteps), the robot manages to successfully (with success rate ~ 0.9 and ~ 0.8) grasp the *006_mustard_bottle* and the *021_bleach_cleanser*. In Fig. 6, we show a successful grasping sequence during the *006_mustard_bottle* training. We refer the reader to the video submitted as supplementary material[3] which shows two successful grasps on the two considered objects and illustrates the whole training process for the *006_mustard_bottle*.

VI. Limitations

With the proposed method, we managed to speed-up the training for a multi-fingered grasping task by a factor of 5, while also removing the need for task demonstrations. However, training a model for grasping a single object for $1M$ timesteps still requires a considerable amount of time, especially if these must be executed on the real robot.

From a qualitative analysis of the residual policies, we noticed that they struggle to react to failures during grasp execution. We believe that this is the reason why their overall success rate is comparable to the open-loop pipeline **Demonstrations**. We plan to further improve the reward function, adding components for adapting the position of the fingers once grasp failure is detected. Moreover, in the current problem setting, the observation of the environment comprises an estimate of the initial position of the object, which is kept constant throughout the grasping episode. We plan to integrate a class-agnostic tracker to keep updating the estimated position of the object. This can help obtaining more reactive grasping policies.

Finally, while we show that RESPRECT does not need a hand-tuned controller, and that this has some advantages with respect to the state-of-the-art, it is fair to say that our method requires a suitable base policy pre-trained with an *Actor-Critic* DRL algorithm, such as SAC [3] or G-PAYN [7].

VII. Conclusion

Grasping with multi-fingered robotic hands is an important task for dexterous manipulation. State-of-the art DRL approaches that tackle this problem suffer from data-inefficiency and are difficult to deploy on real robots. Some recent works propose to train a policy only with simulated data, and to transfer the trained policy on the real robot without any adaptation. However, these approaches are highly dependent on the quality of the simulated environment, and may not be suitable for those cases in which there is a large sim-to-real gap. In addition, these approaches are intrinsically off-line and do not allow the robot to adapt after its deployment. In this perspective, we propose RESPRECT, with the aim of speeding-up the training of DRL policies to grasp novel objects. The proposed approach learns a residual policy for the object at hand on top of a policy pre-trained on a different set of objects. In contrast to existing RRL methods that leverage model-based controllers, we employ a pre-trained policy. This allows to use the weights of the latter to warm start some components of the residual policy to significantly speed-up the training.

We show that RESPRECT achieves a comparable success rate as G-PAYN [7] in a fraction of the training timesteps and without using task demonstrations. Moreover, RESPRECT outperforms two fine-tuning and MetaRL baselines for adaptation of a pre-trained policy on a new target object both in

[3]https://youtu.be/JRsBLVclhpg

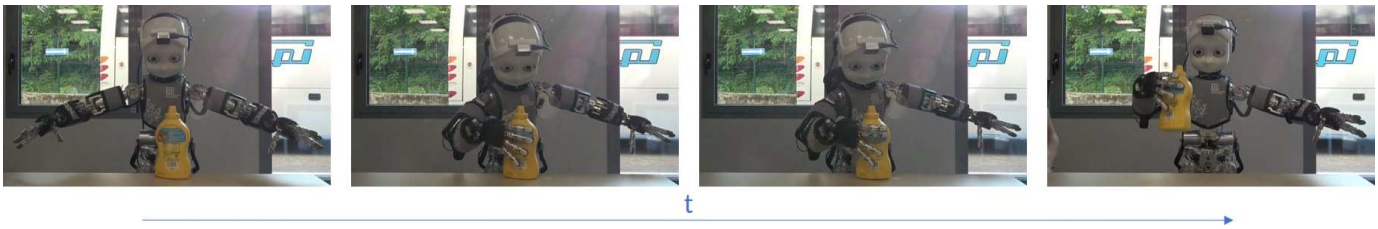

Fig. 6. Object grasping with the iCub humanoid robot. In this exemplar sequence the policy was trained to grasp the *006_mustard_bottle* with RESPRECT on the real robot.

terms of success rate and training steps required to achieve comparable performance.

Finally, we deploy the proposed RESPRECT on the real iCub [6] humanoid, showing that it is possible to obtain a policy that is trained directly on the real robot.

As a future work, we plan to improve the grasping pipeline to obtain policies which are more reactive to failures.

APPENDIX A

In Fig. 7, we compare the success rate achieved by **Fine-Tuning** updating the policies for one and ten gradient steps at each training iteration.

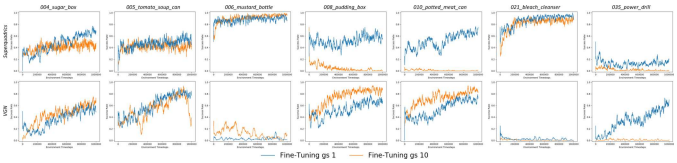

Fig. 7. We evaluate **Fine-Tuning** considering one and ten gradient steps at each training timestep.

APPENDIX B

In Fig. 8, we evaluate **Reptile** updating the policies for one and ten gradient steps at each training iteration.

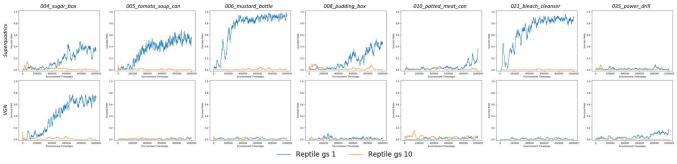

Fig. 8. We evaluate **Reptile** considering one and ten gradient steps at each training timestep.

APPENDIX C

In Fig. 9, we compare the success rate achieved by **G-PAYN** trained with different visual backbones for $5M$ environment timesteps.

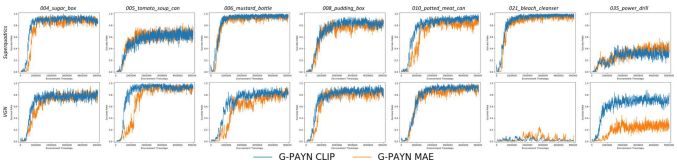

Fig. 9. We evaluate **G-PAYN** trained with different visual backbones (MAE and CLIP) for $5M$ environment timesteps.

APPENDIX D

In Fig. 10, we overview the **Residual** approach used to compare results obtained with **RESPRECT** in Sec. V.

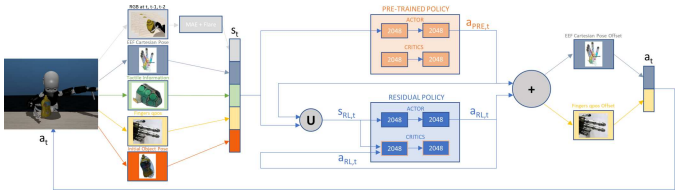

Fig. 10. **Residual** overview. We compute state s_t from RGB images at timesteps t, $t-1$ and $t-2$, end-effector cartesian pose, tactile information and finger joint poses. We compute action a_t (composed of cartesian offsets for the end-effector and finger joint offsets) combining the outputs $a_{PRE,t}$ of the pre-trained policy and $a_{RL,t}$ of the residual policy. Note that $a_{RL,t}$ is the output of the residual policy *Actor*, given the concatenation of s_t and $a_{PRE,t}$ into $s_{RL,t}$. We train only the two 2048-dimensional fully connected layers in the residual *Actor* and *Critics*. For the sake of clarity, we do not report the input of the *Critics* in the pre-trained policy, and the output of both the *Critics* being the same as the ones in SAC [3].

REFERENCES

[1] OpenAI: Marcin Andrychowicz, Bowen Baker, Maciek Chociej, Rafal Jozefowicz, Bob McGrew, Jakub Pachocki, Arthur Petron, Matthias Plappert, Glenn Powell, Alex Ray, et al. Learning dexterous in-hand manipulation. *The International Journal of Robotics Research*, 39(1):3–20, 2020.

[2] Tao Chen, Jie Xu, and Pulkit Agrawal. A system for general in-hand object re-orientation. In *Conference on Robot Learning*, pages 297–307. PMLR, 2022.

[3] Tuomas Haarnoja, Aurick Zhou, Kristian Hartikainen, George Tucker, Sehoon Ha, Jie Tan, Vikash Kumar, Henry Zhu, Abhishek Gupta, Pieter Abbeel, et al. Soft actor-critic algorithms and applications. *arXiv preprint arXiv:1812.05905*, 2018.

[4] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[5] Yuzhe Qin, Binghao Huang, Zhao-Heng Yin, Hao Su, and Xiaolong Wang. Dexpoint: Generalizable point cloud reinforcement learning for sim-to-real dexterous manipulation. In *Conference on Robot Learning*, pages 594–605. PMLR, 2023.

[6] Giorgio Metta, Lorenzo Natale, Francesco Nori, Giulio Sandini, David Vernon, Luciano Fadiga, Claes von Hofsten, Kerstin Rosander, Manuel Lopes, José Santos-Victor, Alexandre Bernardino, and Luis Montesano. The iCub humanoid robot: an open-systems platform for research in cognitive development. *Neural networks : the official journal of the International Neural Network Society*, 23(8-9):1125–34, 1 2010.

[7] Federico Ceola, Elisa Maiettini, Lorenzo Rosasco, and Lorenzo Natale. A grasp pose is all you need: Learning multi-fingered grasping with deep reinforcement learning from vision and touch. In *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2985–2992, 2023.

[8] Giulia Vezzani, Ugo Pattacini, and Lorenzo Natale. A grasping approach based on superquadric models. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1579–1586, 2017.

[9] Tianqiang Zhu, Rina Wu, Jinglue Hang, Xiangbo Lin, and Yi Sun. Toward human-like grasp: Functional grasp by dexterous robotic hand via object-hand semantic representation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 1–14, 2023.

[10] Kelin Li, Nicholas Baron, Xian Zhang, and Nicolas Rojas. Efficientgrasp: A unified data-efficient learning to grasp method for multi-fingered robot hands. *IEEE Robotics and Automation Letters*, 7(4):8619–8626, 2022.

[11] Hongzhuo Liang, Lin Cong, Norman Hendrich, Shuang Li, Fuchun Sun, and Jianwei Zhang. Multifingered grasping based on multimodal reinforcement learning. *IEEE Robotics and Automation Letters*, 7(2):1174–1181, 2021.

[12] Mel Vecerik, Todd Hester, Jonathan Scholz, Fumin Wang, Olivier Pietquin, Bilal Piot, Nicolas Heess, Thomas Rothörl, Thomas Lampe, and Martin Riedmiller. Leveraging demonstrations for deep reinforcement learning on robotics problems with sparse rewards. *arXiv preprint arXiv:1707.08817*, 2017.

[13] Ashvin Nair, Bob McGrew, Marcin Andrychowicz, Wojciech Zaremba, and Pieter Abbeel. Overcoming exploration in reinforcement learning with demonstrations. In *2018 IEEE international conference on robotics and automation (ICRA)*, pages 6292–6299. IEEE, 2018.

[14] Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. 2016.

[15] Aravind Rajeswaran, Vikash Kumar, Abhishek Gupta, Giulia Vezzani, John Schulman, Emanuel Todorov, and Sergey Levine. Learning complex dexterous manipulation with deep reinforcement learning and demonstrations. In *Proceedings of Robotics: Science and Systems*, Pittsburgh, Pennsylvania, June 2018.

[16] Ashvin Nair, Abhishek Gupta, Murtaza Dalal, and Sergey Levine. Awac: Accelerating online reinforcement learning with offline datasets. *arXiv preprint arXiv:2006.09359*, 2020.

[17] Mandi Zhao, Pieter Abbeel, and Stephen James. On the effectiveness of fine-tuning versus meta-reinforcement learning. *Advances in Neural Information Processing Systems*, 35:26519–26531, 2022.

[18] Kate Rakelly, Aurick Zhou, Chelsea Finn, Sergey Levine, and Deirdre Quillen. Efficient off-policy meta-reinforcement learning via probabilistic context variables. In *International conference on machine learning*, pages 5331–5340. PMLR, 2019.

[19] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *International conference on machine learning*, pages 1126–1135. PMLR, 2017.

[20] Tom Silver, Kelsey Allen, Josh Tenenbaum, and Leslie Kaelbling. Residual policy learning. *arXiv preprint arXiv:1812.06298*, 2018.

[21] Tobias Johannink, Shikhar Bahl, Ashvin Nair, Jianlan Luo, Avinash Kumar, Matthias Loskyll, Juan Aparicio Ojea, Eugen Solowjow, and Sergey Levine. Residual reinforcement learning for robot control. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 6023–6029. IEEE, 2019.

[22] Gerrit Schoettler, Ashvin Nair, Jianlan Luo, Shikhar Bahl, Juan Aparicio Ojea, Eugen Solowjow, and Sergey Levine. Deep reinforcement learning for industrial insertion tasks with visual inputs and natural rewards. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5548–5555. IEEE, 2020.

[23] Alireza Ranjbar, Ngo Anh Vien, Hanna Ziesche, Joschka Boedecker, and Gerhard Neumann. Residual feedback learning for contact-rich manipulation tasks with uncertainty. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2383–2390. IEEE, 2021.

[24] Todor Davchev, Kevin Sebastian Luck, Michael Burke, Franziska Meier, Stefan Schaal, and Subramanian Ramamoorthy. Residual learning from demonstration: Adapting dmps for contact-rich manipulation. *IEEE Robotics and Automation Letters*, 7(2):4488–4495, 2022.

[25] Minttu Alakuijala, Gabriel Dulac-Arnold, Julien Mairal, Jean Ponce, and Cordelia Schmid. Residual reinforcement learning from demonstrations. *arXiv preprint arXiv:2106.08050*, 2021.

[26] Andy Zeng, Shuran Song, Johnny Lee, Alberto Rodriguez, and Thomas Funkhouser. Tossingbot: Learning to throw arbitrary objects with residual physics. *IEEE Transactions on Robotics*, 36(4):1307–1319, 2020.

[27] Krishan Rana, Ben Talbot, Vibhavari Dasagi, Michael Milford, and Niko Sünderhauf. Residual reactive navigation: Combining classical and learned navigation strategies for deployment in unknown environments. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 11493–11499, 2020.

[28] Michel Breyer, Jen Jen Chung, Lionel Ott, Siegwart Roland, and Nieto Juan. Volumetric grasping network: Real-time 6 dof grasp detection in clutter. In *Conference on Robot Learning*, 2020.

[29] Wenling Shang, Xiaofei Wang, Aravind Srinivas, Aravind Rajeswaran, Yang Gao, Pieter Abbeel, and Misha Laskin. Reinforcement learning with latent flow. *Advances in Neural Information Processing Systems*, 34:22171–22183, 2021.

[30] Ilija Radosavovic, Tete Xiao, Stephen James, Pieter Abbeel, Jitendra Malik, and Trevor Darrell. Real-world robot learning with masked visual pre-training. In *Conference on Robot Learning*, pages 416–426. PMLR, 2023.

[31] Kristen Grauman, Andrew Westbury, Eugene Byrne, Zachary Chavis, Antonino Furnari, Rohit Girdhar, Jackson Hamburger, Hao Jiang, Miao Liu, Xingyu Liu, et al. Ego4d: Around the world in 3,000 hours of egocentric video. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 18995–19012, 2022.

[32] Laura Downs, Anthony Francis, Nate Koenig, Brandon Kinman, Ryan Hickman, Krista Reymann, Thomas B. McHugh, and Vincent Vanhoucke. Google scanned objects: A high-quality dataset of 3d scanned household items, 2022.

[33] Kevin Zakka. Scanned Objects MuJoCo Models, 7 2022.

[34] Yu Xiang, Tanner Schmidt, Venkatraman Narayanan, and Dieter Fox. Posecnn: A convolutional neural network for 6d object pose estimation in cluttered scenes. 2018.

[35] Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ international conference on intelligent robots and systems*, pages 5026–5033. IEEE, 2012.

[36] Antonin Raffin, Ashley Hill, Adam Gleave, Anssi Kanervisto, Maximilian Ernestus, and Noah Dormann. Stable-baselines3: Reliable reinforcement learning implementations. *Journal of Machine Learning Research*, 22(268):1–8, 2021.

[37] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International Conference on Machine Learning*, pages 8748–8763. PMLR, 2021.

[38] G. Metta, P. Fitzpatrick, and L. Natale. YARP: Yet another robot platform. *International Journal of Advanced Robotics Systems*, 3(1):43–48, 2006.

[39] Ugo Pattacini, Francesco Nori, Lorenzo Natale, Giorgio Metta, and Giulio Sandini. An experimental evaluation of a novel minimum-jerk cartesian controller for humanoid robots. In *2010 IEEE/RSJ international conference on intelligent robots and systems*, pages 1668–1674. IEEE, 2010.

[40] Alex Nichol, Joshua Achiam, and John Schulman. On first-order meta-learning algorithms. *arXiv preprint arXiv:1803.02999*, 2018.

[41] Stephen James, Zicong Ma, David Rovick Arrojo, and Andrew J Davison. Rlbench: The robot learning benchmark & learning environment. *IEEE Robotics and Automation Letters*, 5(2):3019–3026, 2020.