2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
October 14-18, 2024. Abu Dhabi, UAE

Incremental Learning of Robotic Manipulation Tasks through Virtual Reality Demonstrations

Giuseppe Rauso[1], Riccardo Caccavale[1] and Alberto Finzi[1]

***Abstract*— We propose an incremental, modular, and extensible method for learning robotic manipulation tasks using a limited number of demonstrations provided in Virtual Reality, while assuming minimal prior information about the objects to be manipulated. The developed framework enables an incremental training process in which the operator first demonstrates specialized tasks to the robotic system, and subsequently more complex tasks, exploiting the skills learned during the previous phases. We illustrate and discuss the method at work considering picking tasks performed by manipulators equipped with multi-fingered sensorized hands. The experimental evaluation highlights the feasibility and advantage of the proposed method, particularly in terms of modularity, low number of demonstrations, and reliability of the trained system.**

I. INTRODUCTION

In this work, we address the problem of incremental learning robotic manipulation tasks by training the system through demonstrations provided by a human operator in a virtual reality simulation. Learning from Demonstration (LfD) is particularly interesting for robotic task training because, on the one hand, it provides a natural alternative to robot programming, and on the other hand, it allows for expediting other forms of learning by leveraging the guidance provided by human demonstrations. In this setting, the ability to provide demonstrations to the robotic system in a virtualized environment is particularly appealing because it bypasses the complexities of human-robot interaction in a real environment. However, demonstrating a manipulation task in a virtual reality (VR) also presents several challenges due to the gap between the real system and the simulated one, along with the difficulty of recreating, in simulation, the conditions necessary for providing realistic demonstrations to effectively train the robotic behavior. To address these challenges, we propose an incremental approach to VR training from demonstrations that allows the operator to first demonstrate specialized tasks in simple and focused scenarios. Subsequently, more complex and articulated tasks are trained by leveraging behaviors already trained in the previous stages. Our objective is to provide an incremental, modular, and extensible method for learning robotic manipulation tasks using a limited number of demonstrations in a virtual environment, while requiring restricted information about the objects to be manipulated. Another relevant aspect of the proposed approach relies on the use of an affordable virtual reality system to provide demonstrations to the robotic system in a safe and simplified manner. The proposed method combines reinforcement learning techniques with imitation learning methods to achieve not only accelerated training through demonstration guidance, but also behaviors influenced by the expert's dexterity and contextual knowledge.

We illustrate and demonstrate our approach by focusing on grasping and lifting tasks performed by manipulators equipped with sensorized multi-fingered hands, assuming limited information about position and size of the objects to be manipulated. In this scenario, demonstrations are firstly provided to train the grasping behavior of the robotic hand from proximal pose to the target object, with subsequent training involving reaching, grasping, and lifting objects using the proximal grasping capability. The collected results confirm the effectiveness of the proposed approach, demonstrating its potential in terms of modularity, transferability, and reliability of the trained skills despite a limited number of VR demonstrations.

II. RELATED WORKS

Different approaches to learning from demonstration methods in a virtual environment can be found in the literature. Approaches based on learning from demonstration have been proposed to guide learning, reducing complexity and improving efficiency [1], [2] in the presence of more or less complex end-effectors [3], [4]. In this context, experts' demonstrations are obtained in various manners: through robot teleoperation [5], video demonstrations [6], kinesthetic demonstrations [7], [8], or through motion capture [9], [10]. In this work, we focus on demonstrations provided by interacting in VR with a manipulator endowed with a multi-fingered end-effector. A similar problem is addressed in [9], where the authors introduce a technique called *Demo Augmented Policy Gradient* (*DAPG*) that incorporates demonstrations recorded in virtual reality using a motion capture glove, while *Behavioral Cloning* (*BC*) [11] is employed to support learning. However, here the approach is not incremental and modular, while a more complex VR setup is employed to train an anthropomorphic hand. Related to our work, an incremental method for training a robotic manipulator in a virtual environment can be found in [12], which proposes a combination of *Proximal Policy Optimization* (*PPO*) [13] and *Generative Adversarial Imitation Learning* (*GAIL*) [14]. Nevertheless, it is worth to note that this approach assumes a simple two-fingered gripper which can only be opened and closed. This contrasts with the grasp training problem addressed in our

[1]Giuseppe Rauso, Riccardo Caccavale and Alberto Finzi are with Department of Electrical Engineering and Information Technology, University of Naples "Federico II" giuseppe.rauso@unina.it, riccardo.caccavale@unina.it, alberto.finzi@unina.it

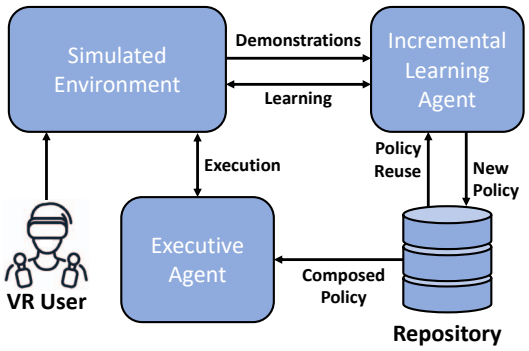

Fig. 1: Overview of the incremental LfD framework

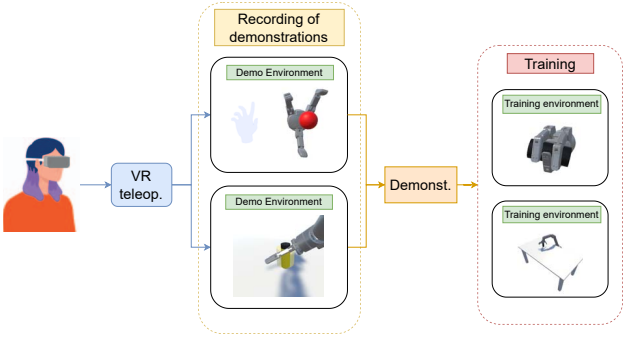

Fig. 2: General pipeline of the system

study, where a more complex sensorized multi-fingered end-effector is deployed.

III. Incremental Learning from Demonstrations

We propose an incremental and modular approach for learning robotic manipulation tasks from demonstrations provided in a virtual reality simulation. The aim is to develop a method that enables to train incrementally complex tasks in simulation, using a limited number of demonstrations and minimal information about the items to be manipulated. The entire framework is illustrated in Fig. 1, wherein the operator provides demonstrations within a virtual environment. These demonstrations are then exploited to learn policies that incrementally handle more complex tasks. During the process, already trained policies can be reused to generate progressively structured ones.

A. Multi-fingered grasping and picking

We illustrate the proposed method at work by considering robotic manipulators tasked to learn how to grasp and pick objects with sensorized multi-fingered hands. In this setting, we describe an incremental training process composed of two stages. The overall pipeline is illustrated in Fig. 2. During the first phase, demonstrations are provided to train the grasping behavior of the robotic hand from a proximal pose (*proximity grasping*) and not accounting for the movement of the robotic arm. In this stage, grasping demonstrations are provided in VR by tracking the human hand movements (Fig. 2, top pipeline). In the second phase, the hand trained during the first phase is exploited to train a more complex task, where the robotic arm is tasked to reach, grasp, and lift a specific object (*grasp-and-lift*). Demonstrations for this task are obtained by teleoperating the robotic arm's end-effector in close proximity to the object, leveraging the assumption that the robotic hand has already been trained to grasp objects from a proximal position (Fig. 2, bottom pipeline). These two training phases are therefore incremental and sequential, as the result of the first phase is exploited for the second one.

B. Simulated Scenario

We consider a simulated robotic setup composed of a *Barrett WAM Arm* manipulator with 7 degrees of freedom equipped with a *BarrettHand* end-effector with 8 degrees of freedom. As for the virtual environment, the virtual reality headset used for teleoperating the robots to obtain demonstrations is a Meta Quest 2. The simulated environments have been developed in *Unity 2021.3.16f1*, while the training was conducted using its *ML-Agents* [15] toolkit which seamlessly incorporates the algorithms used in this work and also allows for a combination of them.

C. Models and methods

The proposed training method integrates imitation learning and reinforcement learning techniques to leverage demonstration guidance along with exploration policies. Specifically, the approach combines Behavioral Cloning (BC), Proximal Policy Optimization (PPO), and Generative Adversarial Imitation Learning (GAIL). While BC enables an initial rapid policy acquisition by mimicking expert actions observed during the demonstrations, PPO is employed alongside GAIL rewards to support policy generalization. PPO works as an on-policy RL algorithm that limits policy updates to avoid too abrupt changes using a surrogate objective function with clipping on the ratio between the old and the new policy. In parallel, GAIL employs a discriminator model (akin to Generative Adversarial Networks) to distinguish between trajectories generated by the learning policy and those from expert demonstrations, and is used to generate an intrinsic reward based on imitation. As training progresses, the influence of BC diminishes, while intrinsic rewards from GAIL enhance imitation of expert behavior, eventually converging BC's to a negligible weight by the end of training. Additional insights about the approach are provided in the next sections.

D. Training Proximity Grasping

The environment for this training phase is inspired by the one proposed in [16]. In this setting, the *BarrettHand* is trained to grasp objects from a proximal position without being attached to an arm or base (see Fig. 3). The hand has a fixed 3D position. At the beginning of each training episode, objects are placed in front of the hand. At the end of the episode, a *force test* is applied to verify the grip. The training objects include cubes, cylinders, and spheres positioned in front of the hand (see Fig. 3) with random position, rotation, scale, and mass within reasonable ranges

suitable for the type of grip. Note that this setup differs from [16], where objects and pre-grasp positions from a dataset [17] are used. In the following, we detail the setup for the learning agent in this scenario, describing observations, actions, reward, demonstrations, and training.

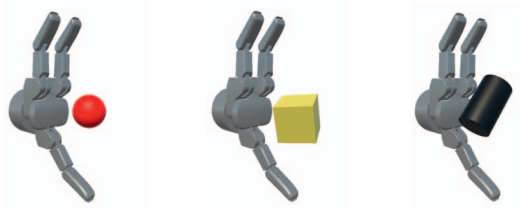

Fig. 3: Training proximity grasping with BarrettHand using spheres, cubes, and cylinders

1) Observations: In our reinforcement learning setup, the agent observes information about the robotic hand and the object to be grasped. As for the robotic hand, we acquire rotations and forces applied on the 8 joints of the BarrettHand. Concerning the objects, we assume limited observable data, including the pose of the associated bounding box (front-top-left and back-bottom-right points, and its rotation), velocity, angular velocity, and the object type. To compensate such limited information, we employ a sensorized gripper with contact sensors (bumpers) installed on the fingertips of the BarrettHand, retrieving contact/no-contact values (1 or 0). This configuration aligns with prior works such as [16], [18], where providing information about contacts with objects has been demonstrated to enhance performance and generalize to objects of varying shapes and sizes. More formally, the agent's observations consist of a tuple $o = (o_c, o_j, o_f, o_{obj}, o_{type})$, where $o_c \in \{0,1\}^3$ represents contact sensors values, $o_j \in \mathbb{R}^8$ denotes the configuration of the hand's joints rotations, $o_f \in \mathbb{R}^8$ are the force measures of the joints, o_{obj} provides observed values about the object, and $o_{type} \in \{0,1\}^3$ is the one-hot vector representing the object type. Here, the observed object values are denoted by the tuple $o_{obj} = (P_1, P_2, q, v, \omega)$, where $P_1, P_2 \in \mathbb{R}^3$ are the front-top-left and back-bottom-right points of the bounding box, $q \in \mathbb{R}^4$ is the object's quaternion, and $v \in \mathbb{R}^3$ along with $\omega \in \mathbb{R}^3$ denote the velocity and angular velocity of the object. All the observation values, except for the values from the contact sensors and the object type, are scaled within the interval $[-1, 1]$.

2) Actions: In the proposed setting, the agent can rotate every joint of the robotic hand on a single axis. Following the observation by [19] that PPO tends to exhibit better performance with discrete actions, we adopt a discrete action space with 3 possible actions assigned to each of the 8 articulations. Specifically, the actions are defined as follows: 0 to keep the rotation unchanged, 1 to increase the rotation, 2 to decrease the rotation. These actions are mapped into joints' movements. For each joint $i \in \{0, \ldots, 7\}$ at time step t, given the action $a_t^i \in \{0, 1, 2\}$ with outcome $d_t^i \in \{0, 1, -1\}$ the new target angle τ_{t+1}^i is then obtained as

$$\tau_{t+1}^i = \tau_t^i + d_t^i \cdot \nu \cdot \Delta t_{\text{fixed}} \quad (1)$$

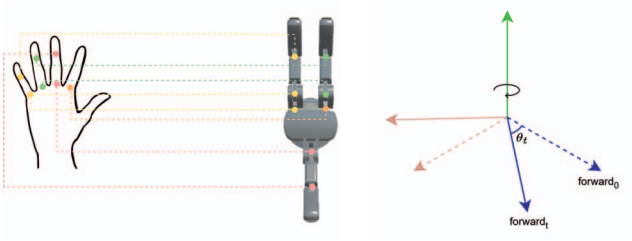

Fig. 4: Mapping of joints from the user's fingers to the BarrettHand (left) and their rotation around the Y-axis (right)

where τ_t^i is the previous value, ν is a fixed rotation speed, while Δt_{fixed} is the fixed time interval.

3) Demonstrations: In the proximity grasping scenario, the operator provides demonstrations by teleoperating in VR the simulated robotic hand exploiting the human hand tracked by the Oculus hand tracking system. To enable this interaction, a mapping between the Oculus hand tracking system and the BarrettHand is needed. Given that the BarrettHand has fewer DoFs compared to the human hand, we designed an interaction schema where a subset of the joints tracked by the Oculus are exploited to control the actuators (see Fig. 4). As four of the tracked joints of the operator's hand rotate around two axes (y and z axes in our representation) these can be mapped into multiple joints of the robotic hand (see lower joint of the little finger in Fig. 4).

Notice that, the hand fingers' joints are controlled through discrete commands, therefore the tracked human hand needs to be suitably translated into such actions. We proceed as follows. At time step t, for each joint of the expert's hand, we consider the rotation angles around the z-axis and the y-axis (if specified in the mapping). We denote these angles as θ_t^i, where i ranges over the number of joints of the robotic hand. The discrete actions associated with such rotation are obtained from the difference between θ_t^i and the target rotation on the corresponding joint mapped on the robotic hand τ_t^i. Specifically, if $|\theta_t^i - \tau_t^i| < \nu \cdot \Delta t_{\text{fixed}}$, the returned action is 0, indicating that the difference between the rotations on the operator's hand θ_t^i and the target rotation on the robot hand τ_t^i is under the minimal discrete rotation. Otherwise, if $\theta_t^i - \tau_t^i$ is positive or negative, the action will be 1 or 2, respectively, denoting that the target rotation needs to be updated to reach the demonstrated movement.

4) Reward: In this setup the reward is defined as follows. After a fixed number of n_{grasp} steps from the beginning of each episode, a *force test* is performed to estimate the quality of the grasp. A positive reward is given if the grasped object withstands a gravity force applied along the main directions, while no reward is earned if the object is lost.

5) Training: For the training described in this section, we recorded only 15 episodes (approximately 3 minutes), specifically 5 episodes per object type. All episodes were successfully completed by the expert. The model used for the policy and the approximation of the value function is a multilayer perceptron (MLP) composed of two fully-connected layers, each one with 128 neurons, and a long short-term

memory (LSTM) layer with a hidden state dimensionality of 64. We use a sequence length of 64 timesteps. As observed in [19], the use of memory via LSTM can improve the agent performance for object manipulation tasks. We rely on implementations of ML-Agents algorithms. For PPO, we adopt a batch size and buffer size of 128 and 16000 respectively, a learning rate of 0.0001, beta as 0.01, and gamma as 0.995. The weight of the reward derived from GAIL is set to 0.01, while the weight of the BC update is set to 0.1 and is annealed over the total number of training steps (i.e., 10 million). Unspecified hyperparameters are left at the default value set by ML-Agents [1]. The training was stopped at 7 million steps, which equates to approximately 5-7 hours (depending on the combination of algorithms used). With the best combination (PPO+GAIL+BC), the agent achieves peak performance after approximately 1.3 million steps, which translates to about an hour of training. The machine used for training in this phase is equipped with an Intel i7-9700K, an NVidia RTX 3070 Ti and 16GB of RAM.

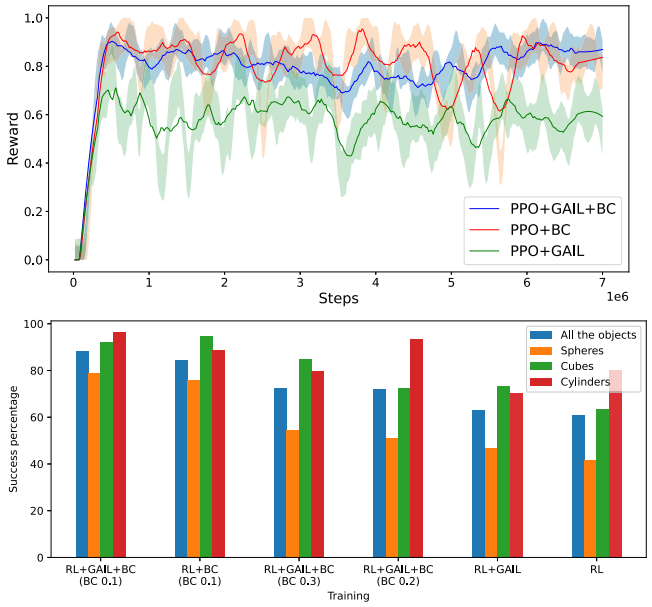

Fig. 5: Reward during training of the 2×128 + LSTM model for the proximity grasping task using various combinations of algorithms (up) along with success percentages obtained during the tests (bottom)

E. Grasp-and-lift with manipulator

In a second stage of learning, the objective is to train the overall robotic system (robot manipulator equipped with the hand) to pick up the objects by leveraging the proximity grasping skill trained in the previous phase. In this case, the primary objective is to train the robotic arm to position its hand in proximity to the target object and to successfully activate proximity grasping. Additionally, the manipulator should also learn how to grasp an object according to a required task (e.g., grasp from the side or grasp from the top). In this setup, the robotic hand is attached to the robotic arm (i.e., the *Barrett WAM Arm*), positioned in front of a table designated for object placement (see Fig. 6 - left). For this specific experimental scenario, we have chosen to exclusively use cylindrical objects, as they allow for various grasping approaches depending on the task at hand. Specifically, we focus on two distinct grasping tasks: a *top grasp*, where the robot approaches the object vertically from above (green zone in Fig. 6 - right-up), and a *side grasp*, where the robot approaches the object horizontally from any side (yellow zone in Fig. 6 - right-down). At the beginning of each episode, the object is randomly scaled and placed on the table in a random position within a defined area (red rectangle in Fig. 6 - left). The agent goal is to successfully execute the grasp, involving hand positioning and proximity grasp activation, in accordance with the required task, such as grasping from the specified side.

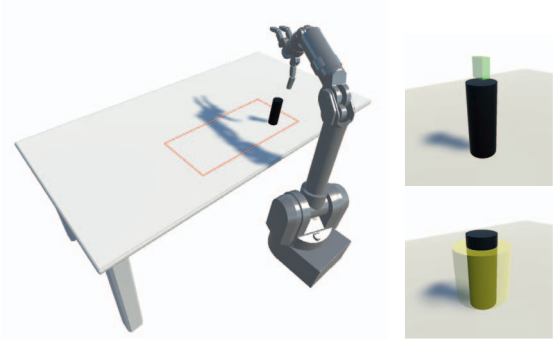

Fig. 6: Representation of the simulated environment for the grasp-and-lift phase (left) along with task-based grasping zones for vertical grasp (right-up, in green) and horizontal grasp (right-down, in yellow)

1) Observations: To move the robotic hand, the agent controls the target position of the end-effector through the inverse kinematics (here, the *Cyclic Coordinate Descent* algorithm is used [20]). The agent thus observes the state of the end-effector, including current position and orientation, target position, wrist rotation, linear/angular velocities. Furthermore, the agent observes the type of task (top grasp or side grasp) and the object's data. Formally, the observed tuple is $o = (o_{obj}, o_{wrist}, o_{EE}, o_{target}, o_{hand}, o_{task})$ where o_{obj} is the observed object tuple (as denoted in Sec. III-D.1), $o_{wrist} = (\theta_{yaw}, \theta_{pitch}) \in \mathbb{R}^2$ is the vector containing the current rotation of the joint responsible for yaw and the current rotation of the joint responsible for pitch of the end effector, $o_{EE} = (EE_{pos}, EE_{rot})$ with $EE_{pos} \in \mathbb{R}^3$ and $EE_{rot} \in \mathbb{R}^4$, the end-effector position and rotation (in quaternion), $o_{target} \in \mathbb{R}^3$ the target position for the IK, $o_{hand} = (v_{hand}, \omega_{hand})$ with $v, \omega \in \mathbb{R}^3$ the hand linear and angular velocities, and $o_{task} \in \{0, 1\}^2$ the one-hot vector representing the task type.

2) Actions: The displacement of the end-effector (i.e., the hand) is accomplished through inverse kinematics. Additionally, the agent can rotate the wrist and activate the grasp

[1] https://unity-technologies.github.io/ml-agents/Training-Configuration-File

to complete tasks. Specifically, the agent has the following branches of actions at its disposal: the displacement of the IK target along the x, y, and z axes, the target pitch and yaw rotations of the wrist, and the activation of the grasp. Similarly to the previous setup, the action space is discrete: actions for target displacement, yaw and pitch rotations can take values of 0, 1, and 2 (representing no change, increase, or decrease). The final action component is a binary value indicating whether to activate the proximity grasp or not.

3) Demonstrations: During the demonstration phase, we employ the controller of the Meta Quest 2 headset to manipulate the target position of the end-effector, to activate the grasp using a button, and to adjust wrist rotations using the stick. Since we are operating in a discrete action space, during the demonstration we determine, for each coordinate, whether to increment, decrement, or maintain the value unchanged depending on the controller's motion. Thus, we calculate the direction vector $d = \text{pos}_{\text{controller}} - \text{pos}_{\text{target}}$ between the controller's position and the target position, and the sign of each coordinate dictates whether to increase, decrease, or leave unchanged the corresponding coordinate of the target position. Similarly, for pitch and yaw rotations, the motion is discretized, and actions are recorded by indicating increments, decrements, or no-changes based on the analog stick's movement. Movement and rotation speeds are fixed.

4) Reward: In this setup, the reward is defined as follows. We assume that the robot has to reach a suitable grasping pose within n_{pos} steps and to activate the grasp. Subsequently, after n_{grasp} steps of proximity grasping, a *force test* is performed, and if passed, a positive reward is given.

5) Training: We recorded 30 episodes for demonstrations (approximately 9 minutes in total), consisting of 15 episodes for top grasp and 15 episodes for side grasp. All of these episodes were successfully completed by the expert. The neural network model has 3 fully-connected layers, each consisting of 512 neurons. By adding a recurrent layer to the model, the agent fails to surpass a reward of 0 after 10 million steps. Consequently, we excluded the LSTM layer from the architecture during training and validation phases. We utilized the ML-Agents toolkit to implement the learning algorithms, with hyperparameters set as those used for proximity grasping training. In this case, the maximum training duration is set to 50 million steps to account for the increased task complexity. The training process took approximately 71 hours on a machine with Intel i5-9300H, NVIDIA GTX 1050 Ti, 8GB RAM.

Training	all objects	spheres	cubes	cylinders
PPO+GAIL+BC (0.1)	**88.5% ±0.5**	**78.9% ±2.6**	92% ±5.3	**96.4%± 1.8**
PPO+BC (0.1)	84.5% ±1.8	75.9% ±4.2	**94.8% ±3.3**	88.6% ±3.5
PPO+GAIL+BC (0.3)	72.5% ±1.3	54.5% ±3.6	84.7% ±3.5	79.9% ±1.7
PPO+GAIL+BC (0.2)	72.2% ±3	51.2% ±9.3	72.7% ±6.1	93.3% ±0.3
PPO+GAIL	63.2% ±2.8	46.9% ±5.5	73.5% ±5	70.4% ±7.2
PPO	60.8% ±3.8	41.7% ±0.9	63.4% ±8.5	80.1% ±0.8

TABLE I: Success rates achieved in the proximity grasping phase with the 2×128 + LSTM model. In parentheses, the weight used for BC (if present).

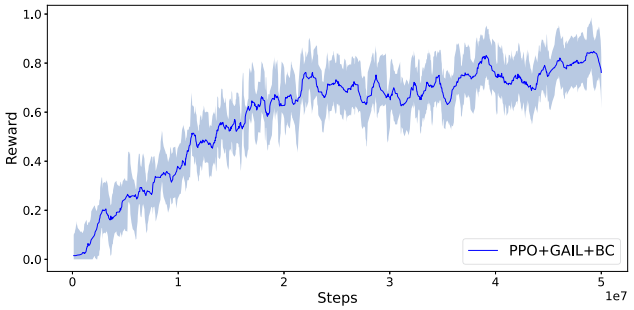

Fig. 7: Reward during training of the 3×512 model for the grasp-and-lift task using PPO+GAIL+BC over 50M steps

IV. Results and Discussion

In this section, we discuss the performance of proposed method for each phase.

A. Proximity Grasping

To evaluate proximity grasping, two tests were conducted. The first test utilized objects used during training, characterized by primitive shapes, serving as approximations of more complex objects. The second test used three objects from a dataset [17]: a bottle, a donut, and a hammer, chosen for their challenging shapes.

For the first test, we compared our proposed combination of RL, BC, and GAIL with alternative combinations (i.e., RL, RL + BC, RL + GAIL, and RL + BC + GAIL with different parametrizations). The proposed method achieves a significantly higher success rate. Results are detailed in Tab. I. Additionally, we evaluated the effectiveness of the recurrent layer by comparing the best combination of algorithms (PPO+GAIL+BC) with or without the recurrent layer, thus with only two fully-connected layers of 128 neurons. The model with the recurrent layer achieves a higher success rate (88.5% vs 82%). We therefore used the former for subsequent evaluations.

In the second test, the agent achieved a satisfactory success rate (about 75.3% ± 1.5) with the novel dataset. It attains a success rate of 66.5% ± 4 for bottles and 71.5% ± 6.9 for hammers, including the "awkward" grasping pose (Fig. 8 - right) obtained when the neck of the bottle or the handle of the hammer faces the palm. Notably, excluding these unnatural grasp positions performance increased to 90% ± 2.2. Another factor taken into account to further test the robustness of the policies is the coefficient of friction of the physical material: for the objects used during training and the previous tests, the value 0.8 was used for both static and dynamic friction coefficients. For real objects, however, the value 0.6 was used. In the tests, the same criterion employed during training is used to define a success. A single test comprises 200 episodes, and the reported values are calculated as the average success rates from three tests performed with different seeds (1241, 824, 3567). The experimental results demonstrate the effectiveness of the policy trained, even when facing realistic objects not encountered during training. This capability supports the feasibility of reusing

the trained hand for proximity grasping across a variety of tasks and objects.

Fig. 8: Tested objects from the dataset [17] (left) and "awkward" position of the bottle and the hammer (right)

B. Grasp-and-lift with manipulator

For the second phase, we calculated the grasp success rate, denoted as $p_{grasp} = n_{succ}/n_{grasp} \times 100$, where n_{succ} represents the number of successful episodes and n_{grasp} the total number of correctly activated grasps. Notice that a correctly activated grasp does not necessarily result in a successful episode, as proximity grasping may fail. In this context, we define a correct grasp activation as the accurate invocation of proximity grasping from the designated grasping position zone (see Fig. 6 - up). The p_{grasp} measure provides an indication about the quality of the proximity grasping in the context of the overall task. The model has been trained for 40 million steps, achieving an overall success rate of 83.8% across both tasks, with 83.7% for horizontal grasping, 83.8% for vertical grasping, and a 95.4% success rate for correctly activated grasps (p_{grasp}). Continuing training for an additional 10 million steps does not yield significantly improved performance, except for horizontal grasping, which increases the success rate to 87.3%. The trained model demonstrates proficiency in learning the tasks, as indicated by the high success rates. Specifically, the percentage of successful activations of grasps (p_{grasp}) further validates the capability to adaptively reuse the learned policy from the first training phase (proximity grasping) across different operational contexts and tasks, thereby enabling modularity and incrementality.

V. Conclusions

We presented an incremental and modular method for learning object manipulation tasks via demonstrations within a virtual reality simulation. We demonstrated this method by focusing on grasping and lifting tasks involving a manipulator equipped with a sensorized multi-fingered hand. In this framework, the hand is initially trained to grasp objects from a proximal position, and subsequently, the manipulator is trained to approach the object and activate the previously learned grasping skill. This setup permits easy demonstration recording for complex manipulators on an affordable platform. The experimental results demonstrate the feasibility and effectiveness of the approach, despite a limited number of demonstrations and information about the manipulated objects. As a future research, we aim to explore the modularity of the approach across various robotic components and increasingly complex tasks.

Acknowledgement

This work was partially supported by the projects: EU Horizon INVERSE (grant 101136067) and euROBIN (grant 101070596), Melody (PRIN PNRR prot. P2022XALNS), SPACE IT UP (PE15 ASI/MUR).

References

[1] P. Sharma, D. Pathak, and A. K. Gupta, "Third-person visual imitation learning via decoupled hierarchical controller," in *Proc. of NIPS*, 2019, pp. 2597–2607.

[2] P. Sermanet, C. Lynch, J. Hsu, and S. Levine, "Time-contrastive networks: Self-supervised learning from multi-view observation," in *Proc. of CVPRW*, 2017, pp. 486–487.

[3] A. Gupta, C. Eppner, S. Levine, and P. Abbeel, "Learning dexterous manipulation for a soft robotic hand from human demonstrations," in *Proc. of IROS*, 2016, pp. 3786–3793.

[4] D. Jain, A. Li, S. Singhal, A. Rajeswaran, V. Kumar, and E. Todorov, "Learning deep visuomotor policies for dexterous hand manipulation," in *Proc. of ICRA*, 2019, pp. 3636–3643.

[5] A. Handa, K. Van Wyk, W. Yang, J. Liang, Y.-W. Chao, Q. Wan, S. Birchfield, N. Ratliff, and D. Fox, "Dexpilot: Vision-based teleoperation of dexterous robotic hand-arm system," in *Proc. of ICRA*, 2020, pp. 9164–9170.

[6] P. Sharma, L. Mohan, L. Pinto, and A. Gupta, "Multiple interactions made easy (mime): Large scale demonstrations data for imitation," in *Proc. of Conf. on Robot Learning*, 2018, pp. 906–915.

[7] M. Vecerík, T. Hester, J. Scholz, F. Wang, O. Pietquin, B. Piot, N. M. O. Heess, T. Rothörl, T. Lampe, and M. A. Riedmiller, "Leveraging demonstrations for deep reinforcement learning on robotics problems with sparse rewards," *ArXiv*, vol. abs/1707.08817, 2017.

[8] R. Caccavale, M. Saveriano, A. Finzi, and D. Lee, "Kinesthetic teaching and attentional supervision of structured tasks in human-robot interaction," *Auton. Robots*, vol. 43, no. 6, pp. 1291–1307, 2019.

[9] A. Rajeswaran, V. Kumar, A. Gupta, J. Schulman, E. Todorov, and S. Levine, "Learning complex dexterous manipulation with deep reinforcement learning and demonstrations," *ArXiv*, vol. abs/1709.10087, 2017.

[10] R. Caccavale, M. Saveriano, G. A. Fontanelli, F. Ficuciello, D. Lee, and A. Finzi, "Imitation learning and attentional supervision of dual-arm structured tasks," in *Proc. of ICDL-EpiRob*, 2017, pp. 66–71.

[11] S. Ross and D. Bagnell, "Efficient reductions for imitation learning," in *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, ser. PMLR, vol. 9, 2010, pp. 661–668.

[12] D. Kawakami, R. Ishikawa, M. Roxas, Y. Sato, and T. Oishi, "Learning 6dof grasping using reward-consistent demonstration," *ArXiv*, vol. abs/2103.12321, 2021.

[13] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *ArXiv*, vol. abs/1707.06347, 2017.

[14] J. Ho and S. Ermon, "Generative adversarial imitation learning," in *Proc. of NIPS*, 2016, p. 4572–4580.

[15] A. Juliani, V.-P. Berges, E. Vckay, Y. Gao, H. Henry, M. Mattar, and D. Lange, "Unity: A general platform for intelligent agents," *ArXiv*, vol. abs/1809.02627, 2018.

[16] H. Merzić, M. Bogdanović, D. Kappler, L. Righetti, and J. Bohg, "Leveraging contact forces for learning to grasp," in *Proc. of ICRA*, 2019, pp. 3615–3621.

[17] D. Kappler, J. Bohg, and S. Schaal, "Leveraging big data for grasp planning," in *Proc. of ICRA*, 2015, pp. 4304–4311.

[18] V. C. V. Kumar, T. Hermans, D. Fox, S. Birchfield, and J. Tremblay, "Contextual reinforcement learning of visuo-tactile multi-fingered grasping policies," *ArXiv*, vol. abs/1911.09233, 2019.

[19] M. Andrychowicz, B. Baker, M. Chociej, R. Józefowicz, B. McGrew, J. W. Pachocki, A. Petron, M. Plappert, G. Powell, A. Ray, J. Schneider, S. Sidor, J. Tobin, P. Welinder, L. Weng, and W. Zaremba, "Learning dexterous in-hand manipulation," *The Int. Journal of Robotics Research*, vol. 39, pp. 20 – 3, 2018.

[20] L.-C. Wang and C. Chen, "A combined optimization method for solving the inverse kinematics problems of mechanical manipulators," *IEEE Trans. on Rob. and Aut.*, vol. 7, no. 4, pp. 489–499, 1991.