

# In-hand manipulation of unseen objects through 3D vision

Martim Freire Pereira  
martimpereira98@tecnico.ulisboa.pt

Instituto Superior Técnico, Lisbon, Portugal

May 2022

## Abstract

Underactuated multi-fingered humanoid hands easily and safely accomplish a wide variety of grasping tasks in human-centric scenarios, questioning about its performance in ordinary manipulation tasks after the grasp of an object. High state-space dimensionality inherent to dexterous multi-finger manipulators poses control difficulties that may be unnecessary in some typical activities, which creates a window of opportunity for cheaper underactuated end-effectors to be employed. We propose a two-stage pipeline system to address in-hand manipulation of an object in a real-world scenario, composed of an *off-the-shelf* category-level object pose estimator to deal with the previously unseen item and a model-free Deep Reinforcement Learning (DRL) algorithm aided by Imitation Learning (IL) to get more robust and natural movements. Our experiments show a positive learning curve for the studied task, dealing reliably with real environment intrinsic problems, as sample inefficiency and noisy object estimations, demonstrating a possible alternative to expensive high Degree of Freedom (DoF) manipulators in some mundane tasks.

**Keywords:** In-hand Manipulation, Reinforcement Learning, Imitation Learning, Pose Estimation, Underactuated Humanoid Hands

## 1. Introduction

Nowadays, where the plan is to bring robots to the comfort of our homes, reliability comes hand-in-hand with adaptability. Elementary parallel grippers, commonly found in assembly lines on industrial plants, do not adequate well to more delicate spaces as human facilities, neither to the tasks demanded of them in these new conditions [2]. Dexterous manipulators, namely humanoid ones, provide higher versatility to a wide range of ordinary tasks, thus increasing the research interest in robotic manipulation over the last couple of years.

New tiny and affordable motors with evolved position and torque sensors allow more and more high-tech manipulators to be available in universities and companies worldwide. Advances in sensory detectors, primarily vision and tactile, are making possible a wide variety of tasks and studies.

Learning techniques lead the software solutions to address manipulation, as their computational needs are being relieved by the increased use of Machine Learning (ML) algorithms in the past years. Nonetheless, too many "trial and error" attempts are required for pure RL methods to produce viable results, thus researchers explore adaptations to the learning algorithms, as exploiting

simulation capabilities, or combining RL with IL.

Craving for super-human robotic dexterity, contemporary approaches still fall short when compared to humans. Our ability to learn new finger poses with just a few tries suggests prior knowledge embedded in us that somehow we are able to dig up and use, speeding up and improving our dexterity faculties. Pick and place in laboratory conditions have advanced significantly in the last years. However, the execution of tool handling actions, such as holding a knife to cut something or holding a pen to write, is not possible to execute with standard grippers. Furthermore, the change of an object's pose within the hand, envisioning the adaptation of the object's pose for a place action, is yet out of reach for such grippers, bequeathing a lot of research to be carried out in this area.

The contributions of this work to the research community are hereby summarized.

1. We present a framework that couples an *off-the-shelf* 6D pose estimation and vision tracking method with a sophisticated learning algorithm for within-hand manipulation in the real world. RL and IL are combined to perform in-hand reorientation tasks with a 7 DoF manipulator.

- II. We gather demonstrations to fine-tune our strategy, highlighting and discussing their impact on training time and goal attainment.
- III. We evaluate the performance of DAPG’s algorithm [9] under non-ideal conditions, namely fluctuations in object pose estimation and slight unconformities in the end-effector actuators value.
- IV. We extend Zhu et al [16] results with more demanding tasks executed by a less controllable manipulator, illustrating the capabilities of RL with IL refinement in accomplishing everyday activities.

**2. Related work**

Initial manipulation approaches usually admit to having the friction coefficients between the object and the hand constant, which is something that does not work for most real-life applications. In the last decade, with the increased usage of Neural Network (NN) driven by big data-associated performance growth, these traditional planning techniques started to become deprecated. Machine Learning has been a top issue in the last couple of years, intrinsically related with tremendous and encouraging results in the robotic manipulation field.

The increase in computer power and simulation performance allows to quickly generate thousands of different experimental data to train these NN using a diversity of learning styles. Tobin et al [11] incorporated this into a revolutionary technique called Domain Randomization (DR), which allowed their real-world object detection and grasping system to be trained solely on simulated RGB images to have a pleasant performance. To the authors’ knowledge, this was the first Deep NN (DNN) capable of bridging the *Reality Gap*, the difficulty of transferring simulated learning experience into real-world practice.

With this technique, learning methods gain a new and thither horizon, strengthening the idea of grasping as means-to-an-end, a requirement for handling objects. Thus, DR [11] allowed OpenAI [8] to follow the natural evolution and present in-hand manipulation of a cube by a 24 DoF dexterous handler, the Shadow hand. A three-camera cage-like structure surrounds the Shadow hand that is used with the palm facing up to provide a supporting surface for the turning of the cube. Memory architectures, as Long Short-Term Memory (LSTM), are crucial building blocks of their DNN, doubling the performance with respect to memoryless networks since most dynamic parameters can not be inferred by a single image. The impressive reorientation exhibits in a real scenario, and the natural rise of human manipulation be-

haviors from training the policy, without any expert demonstrations being provided, support the claims presented in this work, particularly that tactile sensing and real data are not a must for within hand manipulation.

The Rubik’s cube, invented in the mid-seventies, quickly became a top-selling toy puzzle. Countless similar puzzles appeared in the succeeding years, from smaller tow sticker edge Rubik’s cubes to a seven sticker edge and beyond. After a triumphant display of reorientation skills of a cube with a Shadow hand, OpenAI [7] evolved their prior work [8] to be able to solve the original Rubik’s cube. DR [11] passed to Automatic DR (ADR), which gradually increases the difficulty and randomization applied to the cube during training when the same performs well on the current settings, allowing for a smoother transition to the Shadow hand.

Humans tend to grasp objects with a purpose in mind. If stabilization is the main goal, a power grip involving the palm is usually the choice taken. On the other hand, if accurate movements, as gating and pivoting, are required, for instance, to spin the faces of the Rubik’s cube, then a precision fingertip grip is used [3]. However, using the palm of the hand to perform manipulation skills is also a possibility if gravity and contact forces are leveraged [1]. This concept, called extrinsic manipulation, supports the jaw-dropping results of the work of Nagabandi et al [6]. Baoding ball place swap is accomplished in 4h of real data training utilizing a Shadow hand by bringing together improvements in learning dynamic models and online Model Predictive Control (MPC) with a method called Planning with Deep Dynamic Models (PDDM). This unconventional model-based learning approach represents the environment by using deep neural networks in a dynamic, uncertain manner, which mitigates learning techniques’ need for big data and incapability to execute complex tasks.

Another way to mitigate learning techniques’ incapability to execute realistic complex tasks and requiring big data to be successful is to join IL with RL. Rajeswaran et al [9] explores model agnostic DRL in simulation, introducing Demo Augmented Policy Gradient (DAPG) as an extension of Natural Policy Gradient (NPG) with demonstration information bootstraps the exploration challenges. Four tasks (in-hand manipulation of pen, object relocation, tool use, and door opening) are used to evaluate DAPG’s performance against other known methods. Reward shaping techniques, due to the hyperparameter tuning sensitivity, are not enough to help the performance of known Deep Deterministic Policy Gradient (DDPG), while Dynamic Movement Primitives (DMP), although suited for imitation learning, are not able to cope with rich sen-

sory inputs. Zhu [16] employs the DAPG algorithm in three real environment tasks (valve rotation, box flipping, and door opening), showing the positive impact of the demonstrations in the training time and overall task performance.

Real-world environment manipulation tasks demand for an object pose estimation technique. Object pose estimation methods face several challenges, such as viewpoint change, occlusion, clutter, similarities. Whether accomplished by traditional techniques or deep-learning mechanisms, object pose estimation systems fall under two distinguish classes with respect to prior knowledge of the object’s model: instance-level or category-level. Instance-level pose estimators rely on knowing the object’s instance beforehand to improve accuracy and robustness, with the predictor being trained for a distinct type of item. On the other hand, category-level pose estimators make use of the object’s category to make them suitable to work with unknown instances of such class, dealing mainly with intra-class variations and shape discrepancies.

To ease the category-level inherent problem of object representation, Florence et al [4] resort to Dense Object Nets as self-supervised dense object descriptors aiming at visual understanding and manipulation. These task-agnostic NN, guided through 3D vision, show encouraging results with a wide variety of unseen and potentially non-rigid objects, with successful intra-class grasp transfer across 47 hat, shoe, or mug items. Simple adaptations to the network’s training procedure confer easy adaptability from intra-class to inter-class estimation, and the accuracy of the proposed multi-object descriptors resembles the performance of networks that do not distinguish between objects.

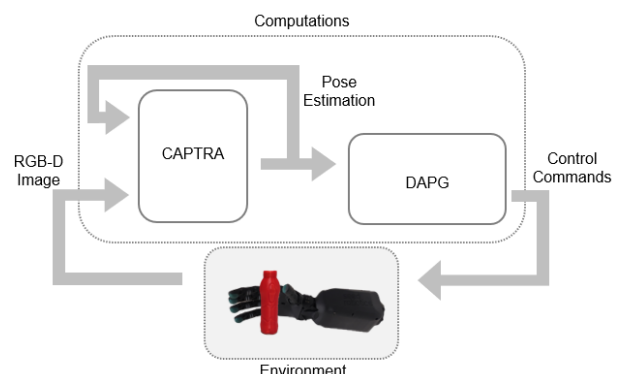
Manuelli et al [5] say that Dense Object Nets[4] are weak to fully represent a class-general configuration-change manipulation task and fail to represent entire object configurations due to self-occlusion. More precisely, they state that “At the foundation of existing pose-estimation methods is the assumption that the geometry of the object can be represented as a parameterized transformation defined on a fixed template.”. Such assumption falls short for large intra-class shape variations, for instance, the task of hanging a mug on a peg. Manuelli et al [5] propose a semantic 3D keypoints method, more robust to shape variations and large item deformations, as an alternative to 6 DoF pose as well as to Dense Object Nets[4]. Their contribution to the research field is KeyPoint Affordances Manipulation (kPAM), a four-stage pipeline comprising instance segmentation, 3D keypoint detection, optimization-based action planning, and geometric grasping action execution. It is up to the

modeler to choose the keypoints, costs, and constraints that encode the task, whit the manual keypoint selection being the Achilles heel this method.

Wang et al [14] train a region-based NN to infer, in their shared canonical representation for all possible object instances within a category denominated Normalized Object Coordinate Space (NOCS), a class label and a instance mask from RGB images, further combining these with depth maps to estimate 6D pose. Wang et al[12] extend their previous DenseFusion [13] network, updating the 2D anchor mechanism to a 3D grid keypoint generator to create their Pose Anchor-based Category-level Keypoint tracker (6PACK). Without manual supervision in the end-to-end learning, 6PACK acquires suitable 3D keypoint representations able to real-time tracking of unseen objects of known instances, outperforming in NOCS [14] benchmark, and accomplishing physical manipulation experiments.

6PACK [12] is the first and only category-level estimation and tracker method known to us designed to and capable of dealing with unknown items besides CAtegorY-level Pose Tracking for Rigid and Articulated objects (CAPTRA) [15]. CAPTRA was the chosen object pose estimation and tracking system present in the first stage of the framework proposed in this article. We recall that we present a framework that couples an *off-the-shelf* 6D pose estimation and vision tracking method with a sophisticated learning algorithm for within-hand manipulation in the real world. A full reading of CAtegorY-level Pose Tracking for Rigid and Articulated objects (CAPTRA) [15] and Demo Augmented Policy Gradient (DAPG) [9] is encouraged for further understanding of details not present in this article.

### 3. Methodology



**Figure 1:** Proposed framework - coupling of CAPTRA [15] with DAPG [9]

The dual-stage system proposed in this article stands as a rather straightforward solution when presented with the problem of controlling an end-

effector that interacts within an environment, either a real or a simulated one. A visual diagram of the general view of the solution can be seen in figure 1, where the task of object in-hand manipulation in a real scenario requires repetition of actuating into the environment, observing the changes, and computing new actions accordingly.

Envisioning the real-world usage of robotic manipulators, testing their viability for operating within our homes in the near future is fundamental. We propose to couple a category-level object pose estimator with a Reinforcement learning algorithm to form our framework, as depicted in figure 1. Our project distinguishes amongst other similar works, as Zhu et al [16], by the adaptation of a learning scheme Rajeswaran et al [9] to encompass non-ideal estimation information.

### 3.1. CAPTRA Adaptations

The first step for using CAPTRA as our object pose estimator and tracker in this project was to strip down everything except the core structure, maintaining the flow as originally designed by Weng et al [15]. We noticed that the object segmentation part, a requirement for the live stream implementation, was not disclosed in CAPTRA’s code. We developed a simple color segmentation algorithm using the OpenCV package to accommodate this need. A color segmentation algorithm benefits from easy code writing at the cost of weaker accuracy when compared to more elaborated techniques. Handcraft-tuning of the filtering values, as well as the requirement for specific hardware in order to be able to operate within acceptable performance ranges, are further drawbacks of such segmentation methodology.

After cleaning the code to salvage the skeleton Point Cloud Neural Networks, we too had to adapt it to our needs, that is, changing experimentally dependent values, as the intrinsics of the camera used to capture the RGB-D image of our object, the bottle. We selected the bottle from categories CAPTRA was pre-trained on, since we thought it could yield the best results for the in-hand manipulation task.

### 3.2. DAPG Adaptations

To shift from simulation to the real-world, we modified the Rajeswaran et al [9] DAPG’s critical aspect, the environment, with respect to both the perception and the actuation. Keeping the training structure, all of Mujoco’s simulation processes, from the static artificial background to the dynamic object and manipulator, are replaced by our Python code. Analogous to the OpenCV Bridge package, our Python code is responsible for bridging the hardware and software at the laboratory by managing the interactions between the setup and the

computations, as illustrated in figure 1. RL’s standard mathematical formalism, the Markov Decision Process (MDP), is a memoryless system characterized by the homonymous underlying rule, the Markov property. It states that past decisions do not influence future state transitions knowing the present state and action. Hence, a MDP is a tuple,  $\langle \mathcal{S}, \mathcal{A}, \mathcal{R}, \mathcal{T}, \rho_0, \gamma \rangle$ , where

- $\mathcal{S} \Rightarrow$  set of all valid states,
- $\mathcal{A} \Rightarrow$  set of all valid actions,
- $\mathcal{R} : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R} \Rightarrow$  reward function,
- $\mathcal{T} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow \mathcal{P}(\mathcal{S}) \Rightarrow$  transition probability function,
- $\rho_0 \Rightarrow$  probability distribution of the initial state,
- $\gamma \Rightarrow$  discount factor.

The control part of our pipeline is modeled as an MDP tuple  $\mathcal{M}$ , where  $\mathcal{T}$  in this model-free framework is unknown. The demonstration dataset groups the state-action pairs with respective reward, per time  $t$  and per trajectory  $i$  in  $D = \{(s_t^i, a_t^i, r_t^i)\}$ . The equivalent on-policy data collected from rolling out the policy  $\pi$  is denoted  $D^\pi$ . The standard definitions of value, Q, and advantage functions are given by

$$V^\pi(s) = \mathbb{E}_{\pi, \mathcal{M}} \left[ \sum_{t=0}^{\infty} \gamma^t \mathcal{R}_t \mid s_0 = s \right],$$

$$Q^\pi(s, a) = \mathbb{E}_{\mathcal{M}} [\mathcal{R}(s, a)] + \mathbb{E}_{s' \sim \mathcal{T}(s, a)} [V^\pi(s')],$$

$$A^\pi(s, a) = Q^\pi(s, a) - V^\pi(s).$$

The concept of an advantage function is important as sometimes there is not the need to classify an action in an absolute sense, but merely a relative evaluation is required. Thus, the advantage function of a particular policy  $\pi$  describes how much better it is to take a specific action  $a$  when in state  $s$  versus randomly selecting the action by sampling the current policy, assuming that after this action one acts according to  $\pi$ .

A good performance of the algorithm relies on selecting the best parameters of the parameterized policy in a way to maximize the sum of the expected reward  $\eta(\pi) = \mathbb{E}_{\pi, \mathcal{M}} [\sum_{t=0}^{\infty} \gamma^t r_t]$ .

In order to make the algorithm more robust, instead of peculiar reward shaping, DAPG incorporates the expert demonstrations into the policy gradient in two phases

- The first phase is a pretraining initialization of the policy with Behavioral Cloning (BC), formulated by

$$\underset{\theta}{\text{maximize}} \sum_{(s, a) \in D} \ln \pi_\theta(a|s). \quad (1)$$

As concluded experimentally by [9] and [16], and detailed by our own experiments in a subsequent chapter of this thesis, a BC policy fails to accomplish the desired tasks due to distribution shift problems.

- The second phase is a fine-tuning of the Reinforcement Learning policy with augmented loss. The details present in the demonstrations are not fully captured simply by initializing the policy with BC, with important aspects remaining hidden in the demonstrations for the RL to explore.

DAPG is built upon NPG with a slight difference in the gradient update. NPG, on the other hand, extends vanilla policy gradient REINFORCE, providing a more stable optimization procedure and faster convergences. REINFORCE gradient, given by

$$g = \frac{1}{NT} \sum_{i=1}^N \sum_{t=1}^T \nabla_{\theta} \log \pi_{\theta}(a_t^i | s_t^i) A^{\pi}(s_t^i, a_t^i, t), \quad (2)$$

with  $N$  being the total number of trajectories and  $T$  representing the time horizon.

This policy gradient version can be achieved from the standard expected return function by some algebraic operations. In summary, one can expand the expectation, move the gradient under the integral and apply the log-derivative trick before returning to the expectation form, and finally express it as a gradient of the logarithm probability. This gives a computable expression for the optimization at hand.

Further improvements can be done to the expression, such as considering merely rewards of future actions, since agents should only reinforce actions based on their consequences. Moreover, an agent should be neutral if the action it takes leads him to an expected result, thus adding a baseline helps stabilize the policy learning and achieve faster results. Several baselines can be used, with the advantage function being one common special case that provides relative information. We recommend the reader to see [10] for more details on these computations.

The gradient is then pre-conditioned with the inverse of the Fisher Information Matrix, computed as

$$F_{\theta} = \frac{1}{NT} \sum_{i=1}^N \nabla_{\theta} \log \pi_{\theta}(a_t^i | s_t^i) \nabla_{\theta} \log \pi_{\theta}(a_t^i | s_t^i)^T. \quad (3)$$

These are then used in the final step of NPG to create the normalized gradient ascend update rule

$$\theta_{k+1} = \theta_k + \sqrt{\frac{\delta}{g^T F_{\theta_k}^{-1} g}} F_{\theta_k}^{-1} g, \quad (4)$$

where  $\delta$  is the chosen step size.

REINFORCE's gradient, in equation (2), is augmented to

$$g_{aug} = \sum_{(s,a) \in D^{\pi}} \nabla_{\theta} \ln \pi_{\theta}(a|s) A^{\pi}(s, a) + \sum_{(s,a) \in D} \nabla_{\theta} \ln \pi_{\theta}(a|s) w(s, a), \quad (5)$$

where the weighting function  $w$

$$w(s, a) = \lambda_0 \lambda_1^k \max_{(s', a') \in D^{\pi}} A^{\pi}(s', a') \quad \forall (s, a) \in D, \quad (6)$$

combines the relevance of the demonstrations towards the policy decisions.

The additional regularization term in (5) provides a reward shaping effect, similar to a trajectory tracking cost, throughout the entire learning procedure since it encourages the policy to be close to the expert actions. The hyperparameters produce a decay of confidence in the expert's demonstrations as the training evolves to avoid bias in the gradient updates. Rajeswaran et al [9] analyses suggest that precise selection of such hyperparameters is not required, but even so, we adopted their values for both lambdas,  $\lambda_0 = 0.1$  and  $\lambda_1 = 0.95$ .

Our action space comprises the 5 motor angle command, while the reward function employed in our experiments, described by

$$\begin{aligned} \mathcal{R} = & - \|\theta - \theta_{goal}\|_2 \\ & + 10 \cdot \mathbb{K}_{\{\|\theta - \theta_{goal}\|_2 < 75^\circ\}} \\ & + 50 \cdot \mathbb{K}_{\{\|\theta - \theta_{goal}\|_2 < 60^\circ\}} \end{aligned} \quad (7)$$

was inspired by the work of Zhu et al [16].  $\theta$  is the angle between the bottle's revolution axes and the horizontal.

The selected values were empirically proposed based on an analysis of the demonstration data collected and the difficulty of the proposed task. It's important to highlight that such a reward shape is only possible due to an approximation we made regarding the available DoF of the bottle when being manipulated. As one can see, the bottle in this experiment is being manipulated in 3D space, not constrained to a 2D plane. However, such conjecture is acceptable as an initial approach to

the problem considering the physical limitations the palm of the hand imposes during the item's handling. This way, as depicted in figure 2, the angle between the bottle's revolution axes and the horizontal, is obtained from a projection of CAPTRA's object pose estimation.

The environment observations comprises 12 values. These are the concatenation of the last two actions taken by the robotic manipulator, each comprising five motor commands, with a couple of values derived from the estimated pose of the object. Incorporating the penultimate actions into the state representation conflicts with the Markovian assumption. This assumption defends that the system obeys to the rule that state transitions depend only on the most recent state-action pair, without requiring prior information to be known. Conscious of such situation, and faced with a considerable number of learning iterations ahead, we opted to include the penultimate actions envisioning a speeding up of the learning procedure by giving more details to the Neural Network. The 6D pose, predicted by CAPTRA, is projected into the plane formed by the palm of the manipulator, originating one of these values. The final value of the state representation is the angle from the current inclination of the object to the goal angle, the desired  $45^\circ$ . This computation is also present as the first term of the reward function depicted above.

#### 4. Experiments



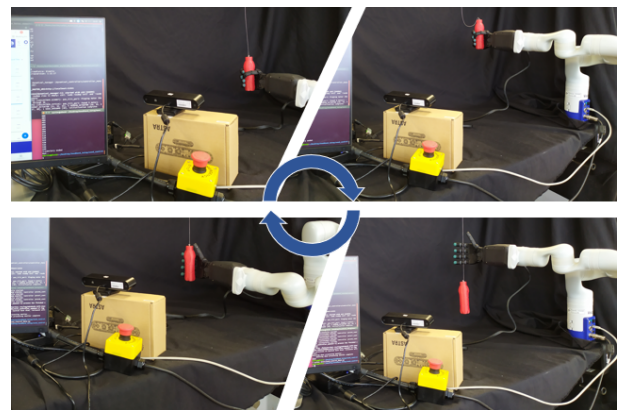
**Figure 2:** In-hand manipulation task explored in this article - initial state (left) and goal state (right)

In-hand manipulation is a recent field of research, with an intensive investigation being conducted as the required hardware starts to be available to companies and universities at affordable prices. The fundamental purpose of our experimental work is to contribute to this research and instigate future studies on this influential unexplored area. Figure 2 illustrates the central task of this project, the RL aided IL teaching an NN to achieve a  $45^\circ$  turn of a bottle using an underactuated hand. On the left, the initial state of our experiments. On the right, we can see the goal target of the task reported and analyzed in this article. The final pose of the bottle should reach a  $45^\circ$  inclination with respect to the vertical axis, as indicated by the

blue annotation in the figure, without falling from the gripper.

For a trial to be considered successful in our experiments, the bottle's angle must be within  $5^\circ$  of the goal state for 5% of the trajectory time. The restrictions imposed on the task constrain the learning of the item's rotation to a pure in-hand manipulation job. In other words, the only motions allowed are accomplished by the underactuated fingers of the manipulator, without any resource for wrist movement, since this would ease the goal-reaching and violate the in-hand premise. Thus, the in-hand premise of this work state that only finger movements can be used to tilt the bottle.

#### 4.1. Setup



**Figure 3:** Experimental environment - The learning cycle comprising: grasping (upper left) the bottle, lifting it up (upper right), executing in-hand movements to tilt the bottle that will eventually drop it (lower right), and re-positioning the hand in a way to start the cycle all over again (lower left)

Figure 3 illustrates the different stages of the learning cycle with actual pictures of the laboratory setup we used during this project. We can observe five key elements for the job, which are

- RGB-D camera - figure 4a
- Handled object - figure 4b
- Supporting arm - figure 4c
- Humanoid end-effector - figure 4d
- Teleoperation glove - figure 4e

The need to have a reset procedure shaped the design of the experiment. Our reset procedure encompasses, on the object side, attaching the 3D printed bottle to the ceiling, and on the manipulator side, connecting it to a robotic arm. A screw on the bottle's lid hooks it to the ceiling by means of an invisible wire. Secured by the non-elastic wire, the drop of the bottle during the learning procedure makes it swing until it eventually hovers in a predefined reset position. Here is where the robotic arm



Figure 4: Experimental material

will place our hand to re-grasp the bottle, lifting it slightly to remove the tension from the wire and re-positioning the hand for a new learning run.

#### 4.2. Demonstration data collection

DAPG algorithm had compelling results in both simulation [9] and real [16] scenarios. The RL-aided IL method uses demonstrations to guide the exploration of the Reinforcement Learning agent in the environment by initializing the NN with a Behavior Cloning technique. In the original work, Rajeswaran et al [9] collected 25 simulated demonstrations for this task, whereas Zhu et al [16] in the real environment collected just 20 trajectories with kinesthetic teaching. We decided to continue the pattern of reducing five demonstrations, taking into account that the collection of such trajectories is quite expensive. Therefore just 15 demonstrations were used in the BC initialization of the final DAPG agent.

Zhu et al [16] resorted to kinesthetic teaching of their manipulators for the collection of the needed demonstrations, whereas we could not make use of the same method due to the particularities of the Seed Robotic hand. A possible solution with the material available in the laboratory was teleoperation. As one can notice, using a glove to control an underactuated manipulator is not an easy job. Moreover, the in-hand manipulation task studied is very complex and hard to complete, requiring a lot of repetitions to achieve 15 reasonable demonstra-

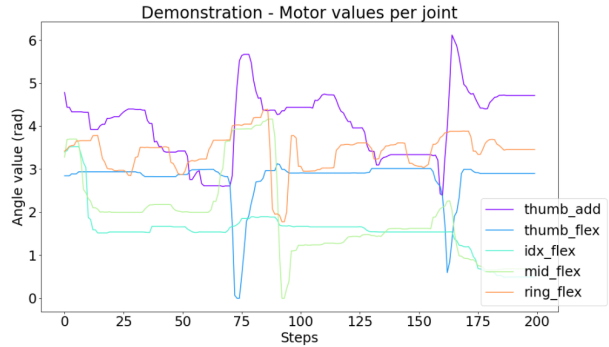


Figure 5: Per-finger motor commands of a demonstration

tions within appropriate time frames.

Figure 5 shows the per-finger commanded motor values of a demonstration. We scaled all our demonstrations to the same number of steps Rajeswaran et al [9] use for their relocation task, 200 steps. We carefully checked that a rescaling of the demonstrations does not change the overall structure of the teaching present within. By this, we mean that the observed shape in figure 5 was equally present in the unscaled version of the same demonstration. We bring the attention of the reader to the thumb spikes exhibited in figure 5, where the thumb trajectory clearly shows a full range of motion characteristic of an upward slide of the bottle within the hand. Learning this motion is crucial for the successful completion of the task proposed once the hardware limitations of the underactuated Seed Robotic hand do not allow for the bottle to tilt  $45^\circ$  unless roughly two-thirds of its size is above the thumb's cavity bump.

#### 4.3. Results & discussion

We created a simple segmentation method to acquire the masks needed for live-stream predictions. For this, we used the capabilities of OpenCV and assembled it to CAPTRA's core Neural Networks. Pure qualitative evaluation of our segmentation method was made at this stage, but we were positively impressed with the results. Nonetheless, an indirect quantitative estimation can be done since the developed method is used in the final experiments.

To get more stable reading for the segmentation, a lamp could have been placed above Astra's camera. However, the white exposure value was extremely high, invalidating this solution. The gamma decay function, described above in section 3, balanced the bottle's exposure for the color segmentation on the one hand. On the other hand, the use of a pose tracker estimator mitigated the influence of the lapses in gathering info from the RGB-D camera.

An uncluttered mask was believed to be crucial for a good estimation of the object's pose. If

so, better segmentation tools would increase the pose's quality and subsequently better learning results for the manipulator. Over the course of experiments, we got the impression this was not a limiting factor, probably due to the fact that the outliers have to survive two checkpoints. After applying to the input RGB-D image the initial mask from the segmentation, CAPTRA computes the centroid of the remaining points. Then it crops out everything outside a sphere with a radius no greater than twice the object's size, which diminishes the influence of segmentation.

We developed a qualitative metric to enrich the results obtained by our experiments. A meaningful movement is defined as a gesture that contributes to reaching the goal in a visible way, as opposed to motions that, for instance, do not tilt the bottle but instead change the fingers in contact with the bottle. We kept the same observer evaluating all the experiments to increase the confidence, within the possible, in this subjective observer-dependent metric.

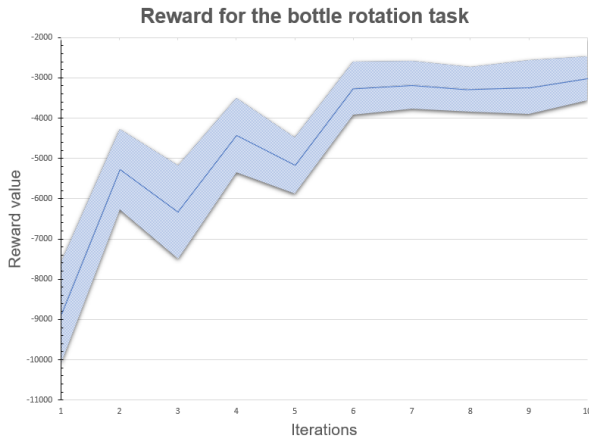


Figure 6: Reward progress throughout the experiments

Aware that complete training of our DAPG agent was not possible under the time restrictions for the delivery of this article, we came to the decision to evaluate the model at each iteration, or epoch, by means of so-called test trials. These test trials, executed after updating the weights of the model in each iteration, complement the reward function cumulative values to indicate if the training is producing improvements in the elaborated actions or if it is a good time to stop the training.

A test trial comprises ten evaluation runs, after which three results are gathered:

- The number of successful runs. A run is considered successful if the bottle's angle is within  $5^\circ$  of the goal angle for 5% of the full trajectory. The design of the task and the Physical restrictions and designing of the manipulation task make sure these bottle angles can only

appear at the end of the trajectory. As no successful run was achieved, this value was suppressed from the tables.

- The mean of the minimum angles reached by the bottle during each run. These are not necessarily the last angle of the bottle before being dropped by the hand, even though, theoretically, should be similar.
- The mean of the last angles reached by the bottle during each run.

The purpose of collecting these values is two-fold. The first is to indicate in a quantitative way if the robot is achieving the goal or not. The second is to reveal if, over the course of each run, the policy tries to tilt the bottle towards the goal angle in a consistent way by checking divergences between the mean angles gathered for each trial test.

Figure 6 depicts the reward function throughout the ten trained and evaluated iterations. The solid line is the mean value with the shaded area illustrating the standard deviation of the return of the paths in each iteration. Referring back to section 3, where the reward function used is explained, we notice that the reward values are negative as a general principle. On top of this, two situations may occur. If the bottle's angle is sufficiently close to the target angle, bonus points are added to the reward. On the contrary, if the bottle is dropped, the reward is aggravated. With this in mind, we can clearly see a rising tendency that shapes in a positive way the learning of our policy.

Our policy's promising results, reflected in the reward function, are corroborated by equally bright marks in the test trials. Despite never getting a successful trajectory, we observe in table 1 a low disparity between mean values for the minimum and the final angles. Considering the precision CAPTRA offers, we argue that the learned policy is understanding well the desired intention of the manipulation task. Moreover, every iteration decreases the mean angles, which verifies, once again, the said claim that the policy is nicely capturing the essential features of the demanded task.

Table 1 shows the collected results. As in any experiment, during the course of the same, situations occur that require amendments to the procedures. In our case, we had to impose a restriction on the amount of discrepancy allowed between consecutive angle values. This requirement appeared halfway through the learning method, as can be seen by the indication next to the fifth iteration. In view of the optimistic results present in figure 6 and table 1, it would be ideal to come up with ways to reduce the unmanageable time required to train the robot's policy. Mechanically speaking, there is not

**Table 1:** Test trials and meaningful movements accomplished per iteration

Iteration	Test Trials		Meaningful Movements
	Mean of minimum angles ( $^{\circ}$ )	Mean of final angles ( $^{\circ}$ )	Number
1	71.32	72.31	1
2	67.04	70.62	1
3	62.89	67.15	1
4	71.15	73.42	2
5 (action clipping)	73.20	75.65	2/3
6 (action clipping)	66.26	67.68	4/5
7 (action clipping)	68.8	70.40	4/5
8 (action clipping)	66.73	68.75	5
9 (action clipping)	66.29	70.14	5
10 (action clipping)	62.55	62.85	5/6

much room to maneuver, as the bottleneck is attached to the physical limitations of the actuator. Nonetheless, this article substantiates preceding research in RL-aided IL as potential control policies for robotic humanoid manipulators.

## 5. Conclusions & future work

In this work, we implemented a model-free Reinforcement Learning algorithm to perform in-hand manipulation of unseen objects by an underactuated humanoid hand, assisted by expert demonstrations and a vision tracking system. The experiments carried out to the test *state-of-the-art* methods from two distinguishing fields, perception and actuation, individually and as a united system to tackle human-centric scenarios. We consider CAPTRA to be pretty solid dealing with unseen bottles, so we think category-level estimators are ready for future challenges. As for DAPG, we corroborated that the challenging collection of demonstrations is the downside of this method. Notwithstanding, in view of our positive learning results, we believe that Reinforcement Learning aided Imitation Learning has a bright future in this field of manipulation.

Our work brings up the simplest form to couple two necessary ingredients for manipulation, perception and actuation. We propose a few directives to carry on the work developed in this project, by first and foremost, continuing the experiments described. The manipulator employed in this work is equipped with tactile sensors that were not used in this experiment. Thus, incorporating them and comparing the results could be important to corroborate the affirmation by [8] or demystify it. In broad strokes, different manipulators are a straightforward addition that future works might pursue. Moreover, more object instances, and classes, would also help assess the possibilities of the framework proposed in this article.

## References

- [1] Y. Bai and C. K. Liu. Dexterous manipulation using both palm and fingers. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1560–1565, 2014.
- [2] A. Billard and D. Kragic. Trends and challenges in robot manipulation. *Science*, 364(6446), 2019.
- [3] J. M. Elliott and K. J. Connolly. A CLASSIFICATION OF MANIPULATIVE HAND MOVEMENTS. *Developmental Medicine & Child Neurology*, 26(3):283–296, June 1984.
- [4] P. R. Florence, L. Manuelli, and R. Tedrake. Dense object nets: Learning dense visual object descriptors by and for robotic manipulation. In A. Billard, A. Dragan, J. Peters, and J. Morimoto, editors, *Proceedings of The 2nd Conference on Robot Learning*, volume 87 of *Proceedings of Machine Learning Research*, pages 373–385. PMLR, 29–31 Oct 2018.
- [5] L. Manuelli, W. Gao, P. R. Florence, and R. Tedrake. kpm: Keypoint affordances for category-level robotic manipulation. *ArXiv*, abs/1903.06684, 2019.
- [6] A. Nagabandi, K. Konoglie, S. Levine, and V. Kumar. Deep Dynamics Models for Learning Dexterous Manipulation. In *Conference on Robot Learning (CoRL)*, 2019.
- [7] OpenAI, I. Akkaya, M. Andrychowicz, M. Chociej, M. Litwin, B. McGrew, A. Petron, A. Paino, M. Plappert, G. Powell, R. Ribas, J. Schneider, N. Tezak, J. Tworek, P. Welinder, L. Weng, Q. Yuan, W. Zaremba, and L. Zhang. Solving rubik’s cube with a robot hand. *arXiv preprint*, 2019.
- [8] OpenAI, M. Andrychowicz, B. Baker, M. Chociej, R. Józefowicz, B. McGrew,

- J. 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. *CoRR*, 2018.
- [9] A. Rajeswaran, V. Kumar, A. Gupta, G. Vezani, J. Schulman, E. Todorov, and S. Levine. Learning Complex Dexterous Manipulation with Deep Reinforcement Learning and Demonstrations. In *Proceedings of Robotics: Science and Systems (RSS)*, 2018.
- [10] J. Schulman, P. Moritz, S. Levine, M. Jordan, and P. Abbeel. High-dimensional continuous control using generalized advantage estimation, 2015.
- [11] J. Tobin, R. Fong, A. Ray, J. Schneider, W. Zaremba, and P. Abbeel. Domain randomization for transferring deep neural networks from simulation to the real world. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 23–30, 2017.
- [12] C. Wang, R. Martín-Martín, D. Xu, J. Lv, C. Lu, L. Fei-Fei, S. Savarese, and Y. Zhu. 6-pack: Category-level 6d pose tracker with anchor-based keypoints. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 10059–10066, 2020.
- [13] C. Wang, D. Xu, Y. Zhu, R. Martín-Martín, C. Lu, L. Fei-Fei, and S. Savarese. Densefusion: 6d object pose estimation by iterative dense fusion. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3338–3347, 2019.
- [14] H. Wang, S. Sridhar, J. Huang, J. Valentin, S. Song, and L. J. Guibas. Normalized object coordinate space for category-level 6d object pose and size estimation. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2637–2646, 2019.
- [15] Y. Weng, H. Wang, Q. Zhou, Y. Qin, Y. Duan, Q. Fan, B. Chen, H. Su, and L. J. Guibas. Captra: Category-level pose tracking for rigid and articulated objects from point clouds. In *Proceedings of the IEEE International Conference on Computer Vision*, 2021.
- [16] H. Zhu, A. Gupta, A. Rajeswaran, S. Levine, and V. Kumar. Dexterous manipulation with deep reinforcement learning: Efficient, general, and low-cost. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 3651–3657, 2019.