

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

Martim Freire Pereira

Thesis to obtain the Master of Science Degree in

Electrical and Computer Engineering

Supervisors

Dr. Plinio Moreno López

Prof. José Alberto Rosado dos Santos Vitor

Examination Committee

Chairperson: Prof. João Manuel de Freitas Xavier

Supervisor: Dr. Plinio Moreno López

Member: Dr. Francisco António Chaves Saraiva de Melo

May 2022

Declaration

I declare that this document is an original work of my own authorship and that it fulfills all the requirements of the Code of Conduct and Good Practices of the Universidade de Lisboa.

Acknowledgments

First, a huge and special thank you to my mother, father, and brother. Their unconditional support over the years is the main reason I'm successfully closing one more chapter, with much more to come. I'm grateful to my grandparents, uncles, and family friends, who always believed in me.

To all my friends, I'm obliged for your encouragement and presence in my life. Thanks for not getting tired of hearing my doubts, your support on the down days was what kept pushing me forward.

Last but not least, I would like to show my gratitude towards my supervisors Plinio Moreno and Prof. José Santos-Victor, for their guidance during the development of this project, to Dimitris Dimou for his weariless patience in thoroughly explaining Seed's hand commands and providing assistance over the entire process, and to Weverton Macedo for his friendly aid recording the demonstrations required to the development of this work.

Resumo

Mãos humanóides de vários dedos subactuadas realizam com facilidade e em segurança uma grande variedade de pegas em cenários desenhados para o ser humano, questionando sobre seu desempenho em tarefas rotineiras de manipulação depois de agarrar um objeto. A alta dimensionalidade do espaço de estados inerente aos manipuladores de vários dedos com mais destreza apresentam dificuldades de controlo que podem ser desnecessárias em algumas destas atividades, o que cria uma janela de oportunidade para que manipuladores subactuados mais baratos sejam empregues. Nós propomos um sistema de dois passos para lidar com a manipulação de um objeto no ambiente real, composto de um mecanismo de estimação de pose de objeto em categoria pronto para lidar com um objecto não antes visto e um algoritmo de Aprendizado por Reforço Profundo sem modelo auxiliado por aprendizagem de imitação para obter movimentos mais robustos e naturais. As nossas experiências mostram uma curva de aprendizagem positiva para a tarefa estudada, lidando de forma confiável com problemas intrínsecos a um ambiente real, como a ineficiência da recolha de amostras e estimativas ruidosas do objeto, demonstrando uma possível alternativa aos caros manipuladores com elevados graus de liberdade em algumas tarefas diárias.

Palavras chave: Manipulação na mão, Aprendizagem por Reforço, Estimativa da Pose, Aprendizagem por imitação, Mãos humanóides subactuadas

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

Contents

1	Introduction	1
1.1	Motivation	1
1.2	Problem formulation & objective	3
1.3	Contributions	4
1.4	Thesis outline	5
2	Fundamental notions, literature review, and key methods	7
2.1	Background	7
2.1.1	Object pose estimation	7
2.1.2	Learning algorithms	9
2.2	Related work	13
2.2.1	Object pose	13
2.2.2	Learning algorithms	18
2.2.3	Dexterous manipulation	22
3	In-hand manipulation methodology	27
3.1	Architecture	27
3.2	Central methods	28
3.2.1	Category-level object pose estimation - CAPTRA	28
3.2.2	Model-free manipulation learning algorithm - DAPG	32
3.3	CAPTRA adaptations	38
3.3.1	Segmentation methodology	39
3.4	DAPG adaptations	40
4	Experiments	43
4.1	Equipment and set-up	44
4.2	Demonstration data collection	55

4.3	Results and discussion	59
5	Conclusion and future Work	67
5.1	Conclusions	67
5.2	Future work	68
A	Appendix A	81
A.1	Continuation of the experiments - short guide	81

List of Figures

1.1	Illustration of the task at hand - A robotic arm (white) coupled with a robotic hand (black) holding a bottle (red), aiming to tilt it 45° (represented situation) using only in-hand movements of the fingers	1
2.1	Reinforcement learning algorithm taxonomy - citations in the text	18
3.1	Proposed framework - general view	27
3.2	CAPTRA differentiable pose tracking pipeline [1]	30
3.3	Result comparison - 6PACK [2](above) and CAPTRA [1] (below) - retrieved from [1]	31
3.4	Proposed framework - coupling of CAPTRA [1] with DAPG [3]	37
4.1	In-hand manipulation task explored in this thesis - initial state (left) and goal state (right)	43
4.2	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)	44
4.3	Experimental environment - structure	46
4.4	RGB-D camera position markers	47
4.5	Orbbec Astra RGB-D camera	48
4.6	Seed Robotic RH8D hand	49
4.7	Seed Robotic - finger nearly rupture situations	50
4.8	Kinova Gen3 arm	51
4.9	Adapter to couple the Seed Robotics hand to Kinova Gen3 arm	52
4.10	Bottle's 3D printing iterations	53
4.11	Immersion's CyberGlove II	54

4.12 Per-finger motor commands of a demonstration	56
4.13 Reward progress throughout the experiments - roughly 100 hours of arm and hand motion.	64

List of Tables

3.1	Pose tracking results averaged over all the six NOCS categories - retrieved from [1]	32
3.2	Comparison of tracking speeds in frames per second (FPS) - retrieved from [1]	32
3.3	Pen in-hand manipulation task - Sample and robot time complexity - retrieved from [3]	36
3.4	Real environment tasks training times in hours - retrieved from [3]	36
3.5	Training parameters	42
4.1	Test trials per iteration	65
4.2	Meaningful movements accomplished per iteration	66

List of Acronyms

6PACK Pose Anchor-based Category-level Keypoint tracker

ADD Average Distance of Model points

ADD-S Average Distance of Model points for Symmetric instances

BC Behavior Cloning

CAPTRA CAtegorY-level Pose Tracking for Rigid and Articulated objects

CNN Convolutional Neural Network

DAPG Demo Augmented Policy Gradient

DNN Deep Neural Network

DoF Degree of Freedom

DR Domain Randomization

DRL Deep Reinforcement Learning

IL Imitation Learning

IRL Inverse Reinforcement Learning

LfO Learning from Observation

MDP Markov Decision Process

ML Machine Learning

NN Neural Network

NOCS Normalized Object Coordinate Space

NPG Natural Policy Gradient

PCNN Point Cloud Neural Network

PPO Proximal Policy Optimization

RGB-D RGB with Depth

RL Reinforcement Learning

TRPO Trust Region Policy Optimization

1 | Introduction

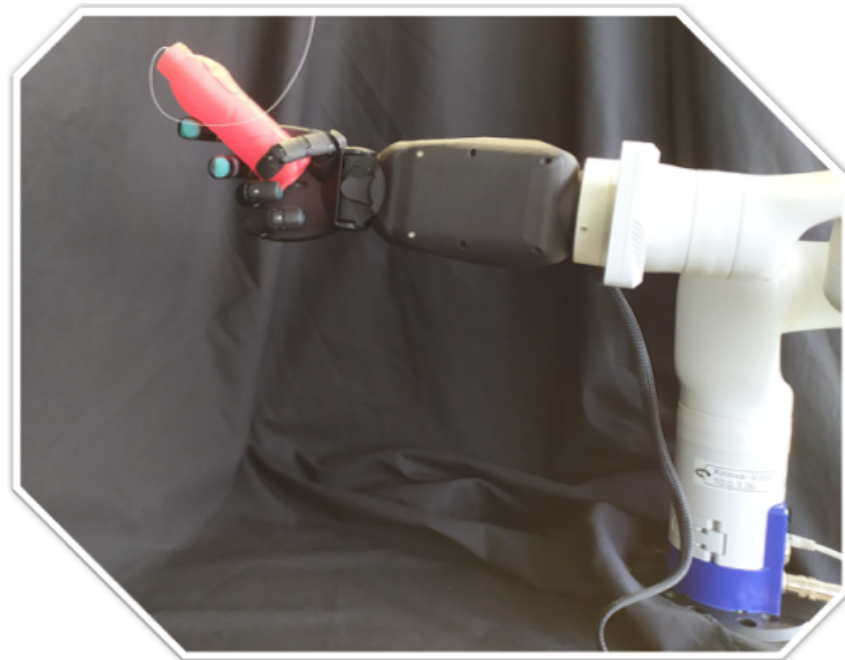


Figure 1.1: Illustration of the task at hand - A robotic arm (white) coupled with a robotic hand (black) holding a bottle (red), aiming to tilt it 45° (represented situation) using only in-hand movements of the fingers

1.1 Motivation

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 [4]. 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.

Some may wonder how object manipulation, a daily and unconscious routine for humans, can pose such a hurdle to robots. Mason [5] answers this by stressing five key aspects: hardware demands (agile yet compact, strong yet sensitive to touch), perception challenges (vision struggles to acquire force data and occlusions are not yet solvable by other sensors such as tactile), control complications (contact forces *per se* are hard to model, combining them becomes an extremely challenging task as motor commands in the presence of multiple contact points require complex hierarchical control models), planning difficulties (enormous state-spaces with non-linear, non-convex, and non-differentiable constraints), and uncertainty management (noise can create deviations from the plan that may lead to task failure).

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 deal with some of the difficulties highlighted by [5], as their computational needs are being relieved by the increased use of Deep Neural Network (DNN) and Machine Learning (ML) algorithms in the past years. These sophisticated algorithms, that mimic the way the human brain operates, aim to recognize relations within their input data. As humans', learning by imitating is a powerful and explored technique. Nonetheless, too many "trial and error" attempts are required for pure Reinforcement Learning (RL) methods to produce viable results, and imitation data required by Imitation Learning (IL) systems is not always available.

Due to these intrinsic issues, some argue that learning is not the ideal way to acquire manipulation skills [4], whereas the majority of the scientific community disagrees and focuses on mitigating them by tweaking the learning algorithms, exploiting simulation capabilities, and 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.

1.2 Problem formulation & objective

Grasping tasks have several published papers over the last few years [6, 7, 8, 9, 10, 11, 12, 13]. Thus, envisioning a full and cognitive manipulation [14] ability by humanoid robots, grasping is just the beginning, a means to an end. Consequently, after-grasp manipulation has a severe impact on achieving a full and cognitive manipulation ability by humanoid robots.

The after-grasp work, often also called in-hand manipulation, comprises the following steps: Given an initial object grasp, where the object is stable to perturbations, the goal is to execute a series of finger limb motions in order to reach another stable grasp. Ideally, visual and tactile inputs provide the necessary perceptual information to decide how to act. More specifically, discover a policy that leads to successful in-hand manipulation of objects, focusing on the rotation of a bottle in the hand and using visual perception to estimate the pose of the bottle with respect to the hand.

The main questions we propose to analyze during this thesis are as follows:

1. Are underactuated five-finger end-effectors sufficiently stable to perform in-hand manipulation tasks?
2. Are category-level object pose estimators steady enough to be incorporated within a framework, in particular an RL one?
3. Can the work by Zhu et al [15] be corroborated, in the sense that Demo Augmented Policy Gradient (DAPG) [3] might be a promising direction for model-free Reinforcement Learning systems beyond simulation environments?

Since the robotic hand has tactile sensors only on the fingertips, our approach uses vision only as perceptual input. We address within-hand manipulation by applying a two-stage pipeline. We employ a category-level perception mechanism, in the first stage, arguing this to be an acceptable choice to deal with unknown object pose estimation. The second stage comprises a learning framework by using an algorithm which does not employ a model of the environment to learn. The control stage uses a model-free Reinforcement Learning plus Imitation Learning.

Recent works have applied RL methods that require a large number of parameters, which means that a large number of samples is needed for the algorithm to find good policies. Therefore, we selected one which counteracts such hurdles with use of IL. Specifically, we selected a recent yet sophisticated algorithm, arguing that its capabilities are not bound to the simulation environment.

The main difference of our work with respect to Zhu et al [15] is that we do not have any way to obtain the ground truth values of the target, we must estimate them by using visual perception. In this sense, our setup is more realistic than the one Zhu et al [15] developed. Instead of having an engineered scenario where the state has access to the actual target, we aim to test a more realistic scenario where external sensors, such as a camera, provide an estimation of the desired target.

1.3 Contributions

This thesis' main objective is to explore real-world in-hand manipulation by humanoid end-effector. A Reinforcement Learning (RL) approach, aided by Imitation Learning (IL), is broached to deal with exploration difficulties underlying the pure RL. With such assistance, the major difficulty shifts to the collection of a sufficient amount of expert demonstrations. Guided by Rajeswaran et al [3], and subsequent experimental work done by Zhu et al [15] in a real environment, we train our model with an considerable number of demonstrations, showing that the combination of RL with IL is a reasonable solution for times to come.

Our contributions to the research community are hereby summarized.

- I. 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. Reinforcement Learning and Imitation Learning are combined to perform in-hand reorientation tasks with a 7 Degree of Freedom (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 [3] algorithm under non-ideal conditions, namely fluctuations in object pose estimation and slight unconformities in the end-effector actuation, that is, inherent lack of precision of the underactuated commands.
- IV. We extend Zhu et al [15] results with more demanding tasks, as in-hand manipulation, executed by a less controllable manipulator, as an underactuated hand with one actuator for all the joints of a finger. This illustrates the capabilities of Reinforcement Learning with IL refinement in accomplishing everyday activities.

1.4 Thesis outline

This thesis has 5 chapters. After the brief introduction to the idea of in-hand manipulation, formulating our specific problem formulation, and pointing expected contributions, the remainder of this document has the following structure.

Chapter 2 presents some fundamental notions on our pipeline procedure together with a suitable literature review. Throughout the chapter, the control part is described after the estimation part, both in section 2.1, where the reader is guided towards relevant background information, as in section 2.2, where an overview of the current *state-of-the-art* methods appear. Section 2.2.3 offers a short in-hand manipulation literature review before, the selected category-level object pose estimator is specified in section 3.2.1, while section 3.2.2 elucidate the reader on more in-depth details of the manipulation algorithm used in this project.

Chapter 3 describes in more detail the elaborated two-stage framework and developed adaptations. The vision stage, in particular the added segmentation technique, is dealt with in section 3.3, while section 3.4 address the control stage refinements.

In chapter 4, our experiments with the humanoid hand are detailed. Section 4.1 enumerates the hardware used and its setup for our tests, with the demonstration acquisition procedures detailed in the subsequent section. In the end, section 4.3 presents the achieved results, analyzing and comparing these results to our theoretical expectations.

Finally, chapter 5 summarizes the work performed, highlighting its main achievements. Moreover, section 5.2 proposes further work directives to extend the activities described in this document.

2 | **Fundamental notions, literature review, and key methods**

This chapter briefly explains the necessary concepts for understanding the work reported in this thesis. In the last couple of years, Neural Networks established themselves as the backbone of almost all techniques available to tackle the long-standing hurdle of estimating the pose of an object. Initially, this and other required background knowledge to comprehend the pose estimation problem are introduced, being later accompanied by relevant *state-of-the-art* literature on the matter.

One of the techniques explored by researchers to deal with the emerging challenge of controlling dexterous five-finger manipulators is to unite two popular learning procedures, Imitation Learning and Reinforcement Learning. This chapter provides insight into each of these methods separately, from some key notations to a few related works.

2.1 Background

In this section, the reader can get familiarised with the key concepts of object pose estimation and control mechanisms based on learning algorithms.

2.1.1 Object pose estimation

For humans, awareness about one's surroundings is crucial to safely and accurately interact with others and navigate the world we live in. As societies begin to support and promote the co-habitation with robots, engineers must ensure these possess sophisticated systems to perceive and understand our world. In particular, handling gadgets or relocating items presupposes an internal knowledge of their whereabouts.

Seen through the glasses of an engineer, especially in the field of robotics, the word *Pose* comprises the localization of an object in the 3D space, that is, its position and orientation with

respect to a reference coordinate frame. While common and cheap measurement instruments, as laser range finders, can precisely determine the location of the object with respect to their reference frame, these can not provide orientation information once such a concept entails a sort of social convention definition.

Pose

One can fully define a position, with respect to a reference frame, by a triplet. Usually designated (x, y, z) , each of these values represents a translation axis-wise in the corresponding unit of measure. Despite living in the same 3D world, orientation has no widely used representation with just three unknowns due to some predicaments with them. The Euler angles convention (roll, pitch, and yaw) suffer from the gimbal lock effect, which is a loss of a Degree of Freedom (DoF) caused by a singularity in the alignment of the angles. The axis-angle convention is inadequate when one needs to take consecutive rotations to the object, as we can not combine two rotations to give an equivalent total rotation. Thus, to unequivocally determine an object's attitude, another two methods are rotation matrices and quaternions, both used in our proposed framework.

A rotation matrix can be seen as a concatenation of three tridimensional unit vectors forming an inertial referential that captures the object's orientation using nine elements. A quaternion does not have a very intuitive definition since it belongs to the set of hypercomplex numbers. As such, its formulation is a linear combination of a real scalar component and a three-element complex part, all orthogonal among themselves $(q_x\mathbf{i} + q_y\mathbf{j} + q_z\mathbf{k} + q_w)$.

Neural Network (NN)

Inspired by the brain anatomy and its way of holding, processing, and transmitting information, a Neural Network (NN) is a collection of associated layers, each containing numerous nodes called neurons. An artificial neuron or perceptron, a mathematical model of the biological one, is the building piece of the network. Each neuron computes a dot product between its internal parameter (weight) and its input. A bias term offsets this result, which is then passed through a non-linear function (activation function). These non-linear activation functions present inside each neuron, which are in turn parallelized within each layer and chained throughout the Neural Network, allow successful learning of very complex mappings between raw input data to the desired outputs by the NN.

To effectively acquire proper modeling of complicated functions, the Neural Network weights must be tuned, learned. In supervised learning, the network is fed with an input for which an expected output is known, for instance, the classification of an object present in a picture. A loss

function, which computes an error between the predicted output and the anticipated one, allows the NN to improve by altering its weights. With a big enough dataset, to avoid bias teaching, the network can theoretically converge to a set of weights that performs well with unseen inputs, as humans do.

The interconnections among the layers of a Neural Network define numerous architectures. A Fully Connected (FC) layer, one of the earliest used, provides an *all-to-all* connection, where each neuron's input is the concatenation of all the neuron's outputs from the previous layer of the NN. More linkage implies more weights in the network, making this format poorly scalable. Further research and innovative ideas led to other powerful NN designs, namely the Convolutional Neural Network (CNN), the current leading method in object detection situations, a subject of great importance in the robotics field.

Convolutional Neural Network (CNN)

Human vision and pattern recognition capabilities are extraordinary due to the way our brain organizes information. Dedicated areas recognize specific patterns, taking advantage of prior knowledge to speed up detection and diminish perception hazards, such as lighting and view-point changes or object's deformation and occlusion.

Image specific CNN's establishing block is a convolution layer, resembling initial regions of our visual cortex. A convolution is a linear dot product multiplication of a matrix of values denominated kernel by an equal size patch of the input image along all the channels (depth) of the input (RGB images have 3 channels, one per color). Specific kernel features are searched by sliding the filter in the image, creating an activation map. Repeating this process for different kernels produces a feature map to the next layer. Pooling layers downsample patches of the feature maps and complement the convolution layers as building blocks of the architecture of a Convolutional Neural Network.

2.1.2 Learning algorithms

In order to understand why merging Reinforcement Learning with Imitation Learning can hold benefits, such as reducing the learning time or an increase in the robustness of the learning procedure, we first need to introduce some fundamental notions.

Reinforcement Learning

As the name suggests, Reinforcement Learning (RL) is a mechanism that teaches by rewarding good performances and punishing bad ones, hoping to foment such desirable behaviors to

the detriment of the poorer ones in the future. RL, and Deep Reinforcement Learning (DRL) in particular, have some specific terminology and formalisms. Even though the in-depth study of concrete algorithms is quite hard and mathematically challenging, key concepts of Reinforcement Learning are rather intuitive.

RL is the "trial and error" learning study of a cyclic interaction between an entity, the agent, and the environment where the agent resides. The agent acts on the environment, perceiving an observation of the state of the world and a reward signal that translates the agent's appreciation of the world's current state. Upon each interaction, the environment can change due to its dynamics or as a result of the accomplished action. One of the major distinctions between families of algorithms in DRL is the number of actions allowed by the environment. An environment can accept an infinite number of different actions to the agent (continuous action spaces) or just a limited set of moves (discrete action spaces).

A state s_t encompasses all of the world's information at any given time t , with the sequence of state-action pairs through time being called a trajectory (also denominated episodes or rollouts in some literature). The observation o_t might be a perfect copy of the state when facing a fully observable environment, or just an incomplete representation of it, a case where the environment is designated by partially observable. A common abuse of notation, in some RL works, is the use of an s where the appropriate technical term should be an o since it's the observation of the state and not the state itself that conditions the action to be taken by the agent.

Commanding the agent's decisions is the policy. A policy π , which can be either deterministic or stochastic, dictates the action to take next depending on the state it finds itself into, or better said, on the agents' perception of the world, striving to maximize the long-term cumulative reward, so called return. The return types distinguish themselves by the reward's time frame, among other features. A finite-horizon undiscounted return sums just on a specific time window, while an infinite-horizon discounted return accumulates rewards forever. The latter also discounts the reward by how far off into the future it is obtained, mimicking human intuition that compensation right now is better than later. Plus, the aforementioned discount factor adds the mathematical benefit of convergence for the infinite sums under reasonable circumstances.

In Deep Reinforcement Learning, the distinction between these reward types fades when compared with traditional Reinforcement Learning. It is quite normal to steer the optimization algorithm towards an undiscounted reward while using a discount factor in the estimation of value functions. Moreover, DRL handles parameterized policies. The before-mentioned policies' outputs are computable functions, described by a set of parameters encoded into weights and biases of a Neural Network (NN) since these can be appropriately tuned through optimization algorithms.

After the high-level introduction, we briefly present some more detailed yet relevant concepts. Whatever the policy and the return measure chosen, the optimal policy π^* is then the policy that maximizes the expected return over the trajectory. Deeply related, the concept of value function $V^\pi(s)$ emerges as the expected return for a particular combination of state-policy pairs. For the initial state s and proceeding actions governed by the policy π , the on-policy value function outputs the expected return for this singular combination. A small twist with the first action taken, not restricting it to the policy, leads us to the on-policy action-value function $Q^\pi(s, a)$. These ideas will be further developed in the upcoming sections when discussing the utilized algorithms.

The above value functions obey so called Bellman equations, a special self-consistency set of equations. The main idea of the Bellman equations is that the value of the initial point is the reward one expects to get from being in that state plus the value of wherever it goes next. Thus, the Bellman equations for the on-policy value functions are given by

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

$$Q^\pi(s, a) = \mathbb{E}_{\substack{a \sim \pi(\cdot|s) \\ s' \sim \mathcal{T}(\cdot|s,a)}} [\mathcal{R}(s, a) + \gamma Q^\pi(s', a')].$$

Before reading the related work, one must be acquainted with RL's standard mathematical formalism, the Markov Decision Process (MDP). As a matter of fact, Partially-Observable Markov Decision Process is technically a more precise way to represent the majority of the learning problems. However, its higher complexity leads researchers towards MDP's simplification. An 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 Markov Decision Process 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.

Imitation Learning

Infants and children learn by imitation, mirroring the grown-ups, querying each one of their five senses for feedback when experimenting with all types of different tasks by themselves. The concept of learning by imitation carried over into the Machine Learning (ML) world, serving as a study basis for hundreds of works nowadays.

In Machine Learning, Imitation Learning (IL) utilizes a set of expert demonstrations, usually coming from humans performing specific tasks, to learn the underlying optimal policy, contradicting RL's process of learning it by the environment's reward signal. Humans can handle tasks where reward signals may only appear way into the future, but machines struggle with this situation, pure RL in particular. Some intensive engineering work is placed into manual tune rewards to soften this issue, yet IL alleviates such burden by acquiring the task expertise directly from the supplied trajectories. Furthermore, it significantly reduces the time required to explore the environment, searching for a solution to a given task.

Expert's demonstrations finality is one way to distinguish the Imitation Learning algorithms, with the loss function employed in the acquisition of the expert's policy being another. The provided specialist's demonstrations can therefore be [16]

- merely mimicked, a process called Behavior Cloning (BC),
- used to correct a given task in the midst of its execution, a process termed Corrective Iterations,
- utilized to deduce the subjacent cost function prior to the underlying policy, a method denominated Reward Inference or Inverse Reinforcement Learning (IRL),
- pixel interpreted to grasp the expert's intention without being specifically told, in a Learning from Observations setting,
- the setup and initialization of Reinforcement Learning algorithms, the way investigated and reported in this thesis.

A thoroughgoing explanation of each of these techniques, along with several study cases, will be presented in the next section. Once again, Markov Decision Process (MDP) is used to formulate IL problems. However, the provided demonstrations that deliver the optimal policy actions that the system will try to discover are the trajectories, and the reward function is unknown.

2.2 Related work

This section describes some of the *state-of-the-art* procedures in the literature.

2.2.1 Object pose

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 lean on object's category to make them suitable to work with unknown instances of such class, dealing mainly with intra-class variations and shape discrepancies.

In literature, object pose estimation articles usually aggregate a 6D indicative as a representation of the number of unknowns present in this task in our 3D world. With respect to input data, Sahin et al [17] review supports the use of color combined with depth information, RGB with Depth (RGB-D), since RGB can provide extra clues to ease detection, thus promoting CNNs as the architecture of choice for 6D pose estimation. If the camera intrinsics are known, both RGB-D or Point Cloud (PC) methods can be used once these camera parameters allow the transformation from one data type to the other. PC points are already referenced to a world frame, while the RGB-D image points are in the camera frame.

Du et al [18] review on object pose estimation techniques distinguishes three core procedures to cope with this task, depending on the characteristics of the intended item. A key feature is the object's texture. A rich texture or strong geometric features suggest that a correspondence-based procedure is a good choice. On the other hand, template-based methods do well in reverse circumstances when the item lacks geometric details or vibrant texture patterns. Lastly, voting-based techniques should be considered when the item is occluded, only partially visible, or when it belongs to a specific category of similar objects. Thus, voting-based routines shall be the selected core procedure of category-level methods to best deal with unseen objects, and template-based systems, in particular, should be avoided because they will have a hard time coping with it [17].

Datasets

Big quantities of diverse and labeled data are required to train recent Deep Neural Network

estimation models. Therefore, the existence of well-known datasets softens these issues and allows further research to appear all around the world. LineMod [19], with 1100+ real frame video sequences of 15 various texture-less objects from several viewpoints, is the most widely used. Along with the dataset, Hinterstoisser et al [19] further introduced an widely used evaluation metric called Average Distance of Model Points (ADD). At the writing time of this thesis, PVN3D [20] is the method with the highest ADD score when RGB-D data is used to train and evaluate the model, and EfficientPose [21] holds the top position in the RGB class. Occlusion LineMod [22], presented in the International Conference on Computer Vision (ICCV) 2015 Occluded Object Challenge, has 20 texture and texture-less objects under three distinct light conditions: bright, dark, and directional spot towards the item. PoseCNN [23] with depth refinement has *state-of-the-art* status in this dataset, with a clear advantage in symmetric objects.

Xiang et al [23] contributions to the field are manifold. They conferred the YCB-Video dataset [23], which compresses accurate 6D object poses of 21 items in 92 videos to a total frame count of 133 827 frames, and extended ADD [19] with ADD-S for symmetric instances. Moreover, their PoseCNN, honed with DeepIM [24], a DNN refinement technique that takes the pose estimation and iteratively matches the rendered with the observed image, owns the best RGB estimation system in the YCB-Video [23] dataset, with PVN3D [20] being the best in the RGB-D class.

More field-specific datasets can be relevant to the task this thesis attempts to solve, particularly hand-object or category-level datasets. Mueller et al [25] created the SynthHands dataset by posing a photorealistic hand model with natural hand motion data, allowing a large variety of genuine hand interactions to be observable in 220 000 RGB-D images. Garcia-Hernando et al [26] built a First-Person Hand Action (FPHA) dataset encompassing 100 000 frames of RGB-D video sequences of 45 daily hand actions with 26 different objects in several configurations. Normalized Object Coordinate Space (NOCS) dataset, developed by Wang et al [27] for category-level 6D pose and size estimation contains 300 000 composited images of which 25 000 for validation, with 18 different real scenes and 42 unique objects.

Core Pose Estimation Procedures

Tekin et al [28] proposed a real-time correspondence-based single-shot detection and pose estimation system. A single-shot detector predicts the bounding boxes for several objects that may be present in the input image in one single pass of the network. It is also known as a one-stage detector because it belongs to a class of object detection models that skip a so-called region proposal stage of previously used two-stage models. A Convolutional Neural Network

regresses 2D projections of the RGB input to 3D object boxes and the Perspective-n-Point (PnP) problem is solved to acquire the estimation of the item's pose. The PnP problem consists of the assessment of the pose of the calibrated camera given a set of n 3D points in the world and their corresponding 2D projections in the image. As RaNdOm SAMple Consensus (RANSAC), it can be seen as a post-processing technique, which previously mentioned EfficientPose, the work of Bukschat and Vetter [21], does not require. Resorting to two subnetworks for translation and rotation prediction, this template-based network can regress 6D poses in a single shot and maintain multiple object detection in a scalable, accurate, and efficient way.

Dataset creators Brachmann et al and Xiang et al also developed template-based estimation systems. Brachmann et al [22] trained a decision forest, sown by a three-component energy function, that computes 6D pose estimations from a single RGB-D image. A depth component deals with normalization and invalid points, an object component deals with in-penetration situations, and a coordinate component deals with deviations from the Forest predictions and ideal object coordinates from the rendered image. Xiang et al [23] developed the well-known PoseCNN, where, using only RGB images as input, they estimate end-to-end 3D translation and 3D rotation separately, in a pipeline manner. It localizes the center and predicts a center distance, after which it regresses the created bounding box to a quaternion representation.

Deng et al [29] template-based PoseRBPF, a Rao-Blackwellized Particle Filter, presents decoupled 3D rotation and 3D translation examination. An autoencoder constructs a feature codebook that allows the discretized rotation to handle symmetries, while depth information can be used to refine the translation part. Outperforming DenseFusion [30] in the RGB-D class of YCB-Video [23], the *state-of-the-art* PoseRBPF can be mixed with PoseCNN (PoseRBPF++) to run solely on RGB information. Deng et al extended their work to visually track objects by fine-tuning its autoencoders to obtain better reconstruction quality [31]. This self-supervised system interacts with the objects to continuously learn and increase performance in estimating 6D poses.

Wang et al [30] end-to-end voting-based Deep Neural Network for 6D pose of known objects, using RGB with Depth color values and Point Cloud geometry information at a pixel level, goes by the name of DenseFusion. Packed with iterative pose refinement, obtains *state-of-the-art* performance in LineMod [19] and YCB Video [23] datasets and completes real-world grasp experiments. Yu et al [32] voting-based pipeline outperforms PVNet and PoseCNN in LineMod [19] and Occluded LineMod [22] benchmarks by leveraging a regularization term denominated Differentiable Proxy Voting Loss (DPVL). This term mimics the hypothesis selection in the voting procedure for more accurate vector-field-based keypoint voting estimation while maintaining the effectiveness in dealing with occlusion and texture-less objects. Training

convergence and performance increase, and minor errors, which caused big deviations from the pose hypothesis in the previous method, no longer do so.

Top ranker in several pose estimation datasets, He et al [20] PVN3D data-driven pose estimation pipeline runs a keypoint-based approach. A deep Hough voting Neural Network for the 3D keypoint detection is followed by Least Squares (LS) fitting, where the authors thought eight key points to be a good trade-off between state-space size and LS fitting requirements. Semantic segmentation is shown to extract global and local features that help predict the 6D object pose. He et al [33] decreased by 13% the number of parameters of their PVN3D and increased the computation speed 2.5x in the successor, Full-Flow Bidirectional fusion network (FFB6D). Local and global feature information from the RGB-D image is extracted by the Convolutional Neural Network (CNN) and PCN (PointCloud Network) simultaneously, with fusion modules facilitating the communication of complementary data between these networks. PVN3D FPS (Farthest Point Sampling) distance only method of selecting the winner estimation of the pose is upgraded with the addition of Scale-Invariant Feature Transform (SIFT)[34].

Visual Servoing and Hand-Object Pose

Manipulation involves a lot of occlusion moments when fingering the target object, thus constantly creating difficulties to pose estimation systems. Calli and Dollar [35] show that a Model Predictive Control (MPC) framework with a visual servoing scheme achieves higher performance and efficiency in the control part of precise tasks, as within hand manipulation, even with rough models of the manipulation process. Lu et al [36] Deep Neural Network can handle self-occlusion and surpass manually selected keypoints thanks to an algorithm specially designed to iteratively optimize keypoint selection. Puang et al [37] keypoint visual servoing framework, KOVIS, comprises two modules, an autoencoder for extracting the key points and a visual servoing Neural Network to learn motions primitives. In virtue of end-to-end simulation training, using adversarial examples, data augmentation, and Domain Randomization [38], their method can be deployed in the real world without requiring precise camera calibration.

Cruciani et al [39] proposes a goal-oriented classification of task success mainly focused on the gripper and item as one, considering that tasks' completion is intrinsically connected to both and not to the object by itself. Merging object and hand pose estimation may additionally help to tackle the occlusion difficulties of within hand manipulation. Tekin et al [40] end-to-end framework, trained on single RGB images, estimates 3D human hand and object poses. Besides, it recognizes the action (as pouring a glass of milk) being performed with *state-of-the-art* performance in less know benchmarks. Oberweger et al [41] implement a feedback loop to improve CNN's mistakes in estimating 3D human hand pose, with or without an object in

hand. Depending on depth images only, their method is close to *state-of-the-art* performance for individually hand pose estimation but outperforms for hand-object jointly. Doosti et al [42] propose HOPE-net, an end-to-end graph-based model trained in FPHA [26] to estimate hand and object pose in real-time from single RGB images.

Category-level methods

To ease the category-level inherent problem of object representation, Florence et al [43] resort to Dense Object Nets as self-supervised dense object descriptors aiming at visual understanding and manipulation. These task-agnostic Neural Network, guided through 3D vision, display 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 [44] say that Dense Object Nets [43] 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 [44] propose a semantic 3D keypoints method, more robust to shape variations and large item deformations, as an alternative to 6DoF pose as well as to Dense Object Nets [43]. 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.

Gao et al [45] add dense geometry to kPAM [44], as a Point Cloud mesh, having in view solving some of its downsides, as the lack of reason over physical properties such as visibility, collision, and grasp stability. They present a fully Convolutional Neural Network that predicts 3D volumetric occupancy from RGB-D input, with Rapidly-exploring Random Tree (RRT*) as a slow but globally optimal planning algorithm. Keypoint selection, manually done, is the Achilles heel of both kPAM versions.

Wang et al [27] train a region-based Neural Network 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 [2] extend their previous DenseFusion [30] 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 [27] benchmark, and accomplishing physical manipulation experiments.

6PACK [2] 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) [1], the object pose estimation and tracking system present in the first stage of the framework proposed in this dissertation.

2.2.2 Learning algorithms

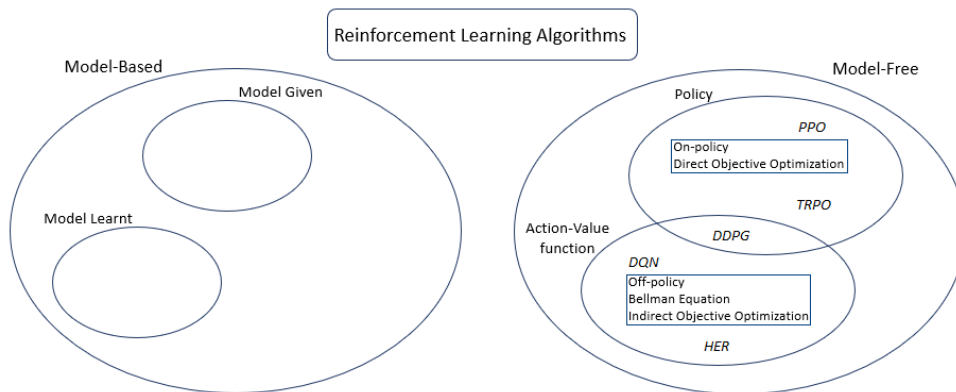


Figure 2.1: Reinforcement learning algorithm taxonomy - citations in the text

Reinforcement Learning

Reinforcement Learning algorithms envision the maximization of cumulative reward obtained by the agent’s action within the environment it inhabits. To select such actions, the agent may, or may not, have access to a model of the world. This distinction, see figure 2.1, is the core difference among RL algorithms, which can be model-based, when a model is given or learned by the agent, or model-free, when such does not happen [16].

To the RL agent, possessing a model of the environment is, considering the above MDP formulation, owning the knowledge of the reward and transition probability functions. As one may notice, having this information available allows planning ahead, making the sampling process

more efficient. Zhang et al [46] propose to interleave policy and model learning, defending it grants better learning speed.

Unfortunately, it is rare, if not impossible, to obtain reliable models of the environment in real scenarios. The agent can attempt to learn the world's model entirely from experience, without guarantees that such learned model actually comprises the essential features of the real environment. Such bias in the model presents a challenge that may lead to sub-optimal solutions, making the easier implementable model-free algorithms a favorite among researchers.

Nagabandi et al [47], wanting the best of both worlds, try model-based plus model-free versions. To take advantage of the better sampling efficiency of model-based, Nagabandi et al employ a medium size Neural Network with Model Predictive Control (MPC) to initialize the model-free network and guide its exploration.

Figure 2.1 illustrates the two main strategies for model-free, with respective bullet point characteristics for each of the Policy Optimization and Q-learning families. Mnih et al [48] pioneered Deep Q-Networks(DQN), a Reinforcement Learning Deep Neural Network that approximates the optimal action-value function, also known by Q-function, for discrete-action model-free. This ground-breaking work propelled other DRL off-policy methods in years to come.

One known example is the work of Lillicrap et al [49], the Deep Deterministic Policy Gradient (DDPG). This model-free actor-critic algorithm is an adapted version of Deep Q-Learning to the continuous action domain, solving most Atari games in 20x fewer steps than DQN [48]. Bellemare et al[50] introduced Arcade Learning Environment (ALE) as the gold standard in Reinforcement Learning benchmarks, containing several Atari games. Nonetheless, generalization and overfitting issues are sometimes present due to the fact that the same environments are both used in training and testing. Cobbe et al [51] initial approach to mitigate these issues was CoinRun, later submitting Procgen benchmark [52] as an attempt to improve the generalization of ALE [50]. The before-mentioned results leave great hopes for the future of DDPG [49], despite needing a more extensive number of training episodes.

Kroemer et al [16] point out another noteworthy contrast among RL algorithms, intrinsically connected to what data is gathered and employed in training. The on-policy update uses data collected by the agent when it is acting according to the most recent version of the policy. On the other hand, the off-policy update makes use of any data obtained during the entire training procedure, regardless of how the agent was exploring the environment when the data was gathered. Off-policy practices, resembling the advantages of model-based over model-free algorithms, prevail in sampling efficiency. Such gain, however, comes at a cost since there are no mathematical guarantees of an optimized policy by improving the performance over

Bellman's equations.

Zhu et al [53] offers an analysis of the future of Reinforcement Learning directed to the sustainability of the method in real-world environments. Learning from raw sensor input, unsupervised self-assignment of rewards, and continuously learning without the need for human resets are the, in their view, three key aspects to have in mind. The high sample complexity of state-of-the-art methods TRPO [54] and PPO [55] discourages their use, while an increase in the variance of initial states instead of trying to reduce it is advised to mitigate the need for human resets within training.

Unsupervised learning, in the context of real-world reset-free vision-based environments, is shown to aggravate the learning procedure instead of simplifying it. Zhu et al [53] concluded that coupling it with a perturbation controller would entice the exploration of unvisited states. Fu et al [56] Variational Inverse Control with Events (VICE), which enforces the maximization of the probability of an event to replace the need for experts demonstrations in reward engineering, is also part of the suggested method, despite struggling with the reset-free idealism. Duan et al [57], inspired by humans' fast learning styles, carry Reinforcement Learning towards meta-learning, aiming at the learning process itself. General-purpose RL algorithms as TRPO [54], with generalized advantages estimation for variance reduction, guide the fast learning agent's training process, the latter modeled by a Recurrent Neural Network (RNN), equipped with Gated Recurrent Units (GRUs).

Imitation Learning

Behavior Cloning (BC) is the simplest form of Imitation Learning. Each state-action pair is interpreted as an independent and identically distributed (i.i.d) example, and the application-specific loss function is minimized in a supervised manner. One can easily see this iid approximation might generate a few issues, particularly since this simplistic assumption conflicts with the previously-mentioned Markov Decision Process formulation, where the state-action combination originates a new state.

It is natural that every model occasionally commits some small mistakes. These tiny deviations from the expert's course can leave the BC agent in an unknown position where it was not trained on, without any expert guidance, not knowing what to do, which could lead to task failure as the errors add up along the way. Thus, Behavior Cloning is likely to fail if long-term planning is required or if the experts' demonstrations one has do not cover the state space of the application well enough.

Direct Policy Learning (DPL) or Behavior Cloning with Corrective interactions [16] is a technique that mitigates BC drifting vulnerabilities by continuously providing more expert tra-

jectories to cases queried by the agent upon execution of an action. Few are the situations where this method can adequately be applied. The need for a full-time expert presence and the vast number of state space configurations are severe drawbacks of this technique. Inverse Reinforcement Learning (IRL), or Reward Inference, is a technique that attempts to learn a reward function from the provided demonstrations and later uses it to find the optimal policy. The optimal policy is, by definition, the expert's one. Therefore, instead of mimicking low-level actions, the idea is to capture the intended movement and then translate it to a reward, later iteratively optimizing the policy for such reward.

Easing the share of rewards between robots when these do not depend on the agent's kinematics, the indirect embodiment in the reward of features challenging to hard-code in the policy, and possible outperformance of the human in some cases are among IRL strongest suits. On the other hand, the aforementioned translation is within this method's weaknesses, being a multiple-solution problem since a set of demonstrations can be the result of various source policies, and there's no way of knowing which is the expert's one.

Appropriate reward types and parameters are an apriori assumption that contrasts algorithms. Abbeel and Ng [58] assume a linear reward function of known features that works well when sufficiently rich features are available. In the absence of such, the use of non-linear reward assumptions, at the cost of more training material, essentially demonstrations, is advisable. Both Finn et al [59] as well as Ho and Ermon [60] rely on alternating reward optimization with policy step refinement as a solution for the experience gathering problem inherent to complex tasks with unknown state transitions (model-free).

Ho and Ermon [60] Generative Adversarial Imitation Learning (GAIL), as a merge of Imitation Learning with Generative Adversarial Networks (GANs), is capable of dealing with high dimension environments thanks to a sharpened version of standard IRL data sampling mechanism that allows the acquisition of the policy more efficiently. Finn et al [59] NNs solve the Inverse Reinforcement Learning multiple-solution problem by approximating a maximum entropy algorithm linked to an overfitting regularization term, while Ramachandran and Amir's [61] proposal is a bayesian learning formulation that uses modified Markov Chain Monte Carlo (MCMC) to sample from the distribution over all possible reward functions given the demonstrations. The latter surpasses previous heuristic approaches, while the first, united with kinesthetic teaching and vision feedback, succeed in real scenario pouring and dish placement tasks.

One can expand the concept of demonstrations, removing the knowledge of the state-action pairs the human expert took present in them and learning strictly from the raw pixels of each image, known as Learning from Observation (LfO). Sermanet et al [62] self-supervised time-contrastive networks leverage different viewpoint training from videos to capture the intended

task. Their unsupervised network learns to identify common behaviors in different-looking images (different viewpoint) and differences in similar-looking images (same viewpoint). Invariance to background and lighting conditions, occlusion, and perspective emerge from the data gathered, teaching the network to focus exclusively on time-changing scenarios. Liu et al [24] experiments show another way to tackle viewpoint invariance in demonstration videos with a context-aware translation model.

Torabi et al [63] lengthen the scope of Behavior Cloning by learning it from observations (BCO), where their agent, trained in a self-supervised fashion, predicts the actions for the state-only demonstrations and performs imitation learning. Simulation results of BCO show comparable performance with GAIL [60] and other *state-of-the-art* methods. Monteiro et al [64] augmented BCO, creating ABCO, where they append a Self-Attention module to both the inverse dynamic model and the policy model. The latter helps the policy to look broadly at states, allowing faster learning in higher dimensional states with a more gradual success rate within iterations.

Extreme applications of Learning from Observation aim to understand the goal from a few images instead of full trajectories, introducing Imitation Learning to meta-learning, the ability to learn how to learn [65, 66].

2.2.3 Dexterous manipulation

The desire for robotic dexterity is not new. Wishing for such dexterity to be present in humanoid robots causes research on how humans accomplish such manipulations themselves. Back in the eighties, the work by Elliott and Connolly [67] already comprised a thorough analysis of the hand movements and examples of characteristic movement patterns. For instance, the utilization of the thumb and the number of fingers involved in each gesture categorize them into distinct classes.

The *state-of-the-art* hardware material available by the end of the century falls short of today's, limiting the experimental setups. Initial 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. Furthermore, prior knowledge of the object's shape is required in advance, constraining the overall usage of such methods. Nonetheless, Farooqu et al [68] were able to apply a priori model knowledge control in the rotation and pivoting of a polygonal shape, studying grasp error and respective recovery strategies in a four-finger spider gripper with integrated force-torque sensors and a palm laser range finder. More modest in terms of hardware were the projects of Han and Trinkle [69] along with Cherif and Gupta [70]. Han and

Trinkle [69] used nonholonomic motion planning and contact kinematics to reorient a sphere in a three-sphere manipulator, while Cherif and Gupta [70] shifted this controller to the simulation environment and adopted a local graph search planner in the configuration space to get quasistatic reorientations of an ellipsoid. Algebraic and geometric reasoning [71] as well as hybrid discrete-continuous dynamical closed-loop models [72] complemented a diversity of routines to deal with the early stages of robot manipulation.

In the last decade, with the increased usage of Neural Networks driven by big data-associated performance growth, these traditional planning techniques started to become deprecated. Even so, other Machine Learning techniques are still able to show promising results, despite their complex cost functions. Calli et al [73] present the first work, to the authors' knowledge, able to write perceptible letters just using within hand motion of the pen on a Yale Openhand T-42 manipulator. Assuming a constant stable contact that disregards slippage, their path planning cost function is based on the Dynamic-Time Warping algorithm to minimize the area between the desired and followed path, with a risk factor regularizer that encapsulates the inaccuracies of the models and forces sensors. Knowing if the object is going to slide, be dropped, or stay stuck if the hand keeps moving the object in the commanded direction [74] are important notions to have in mind. A Yale Openhand model O was the choice of Morgan et al [75] to the same challenging task of letter writing, where the pen is replaced by a QR code marked object. They solved the under-actuated situations using an energy-based Random Forest Regressor model, robust to parameter estimation inaccuracies and adaptable to slippage circumstances thanks to a specially designed four-camera cage vision system.

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 Neural Network using a diversity of learning styles. Tobin et al [38] incorporated this into a 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 great performance. To the authors' knowledge, this was the first Deep Neural Network 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 horizon, strengthening the idea of grasping as means-to-an-end, a requirement for handling objects. Thus, DR [38] allowed OpenAI [76] 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 natural rise of human manipulation behaviors 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. OpenAI [77] evolved their prior work [76] to be able to do the original Rubik's cube. Domain Randomization [38] passed to Automatic Domain Randomization (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. Moreover, in order to easily incorporate new observations and share embeddings between the value and policy networks for concurrent inputs, their Asymmetric Actor-Critic architecture switched the concatenation of the inputs by an embed-and-add approach, where each type of observation is, without any weight sharing, embedded into a latent space that is added later on.

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 [67]. However, using the palm of the hand to perform manipulation skills is also a possibility if gravity and contact forces are leveraged to one's advantage [78]. This concept, called extrinsic manipulation, supports the results of the work of Nagabandi et al [79]. 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 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. It then resorts to optimal control planning to select the best actions to make instead of using the model predictions to learn a policy. To the authors' knowledge, this is the first demonstration that deep dynamic models can be sample efficient and discover fine motor skills with high-dimensional manipulators.

Another way to mitigate learning techniques' incapability to execute realistic complex tasks and requiring big data to be successful, seen in more detail throughout this thesis, is to join Imitation Learning with Reinforcement Learning. Rajeswaran et al [3] explores model agnos-

tic Deep Reinforcement Learning (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. Deep Deterministic Policy Gradient (DDPG) presents the worst success rate of all tested methods. Reward shaping techniques, due to the hyperparameter tuning sensitivity, are not enough to help the performance of DDPG, while Dynamic Movement Primitives (DMP), although suited for imitation learning, are not able to cope with rich sensory inputs. Zhu [15] 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.

3 | In-hand manipulation methodology

This chapter addresses the coupling of the object pose estimation within the learning algorithm in order to form the two-stage framework suggested and presented in this thesis. The first section presents a general view of our in-hand manipulation system, followed by the needed adaptations conducted on both the vision and control processes to incorporate them.

3.1 Architecture

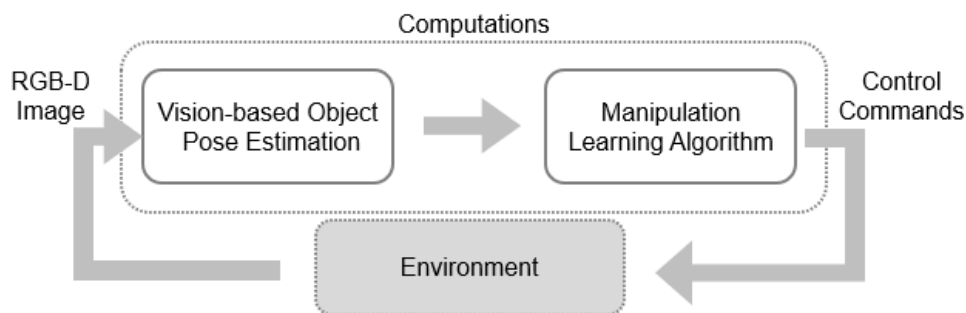


Figure 3.1: Proposed framework - general view

The dual-stage system proposed in this thesis 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 3.1, 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.

3.2 Central methods

This section dives deeper into the selected methods for the two stages of the framework suggested and presented in this thesis. Foremost the vision system is explored, detailing the notation, architecture, and relevant results. Afterward, an equivalent analysis is performed for the control system. A full reading of CAtategory-level Pose Tracking for Rigid and Articulated objects (CAPTRA) [1] and Demo Augmented Policy Gradient (DAPG) [3] is encouraged for further understanding of details not present in this section.

3.2.1 Category-level object pose estimation - CAPTRA

The first part of the proposed framework is the detection and estimation of the pose of the object to manipulate. For this, we resorted to one of the humans' most important senses, the vision. It is noteworthy to highlight the existing correlation of proper gaze usage with higher task success [80, 14]. Despite having access to sensory information from the manipulator, we ignored the tactile feedback provided by the hand as a matter of simplicity.

Object pose estimation, as detailed in the previous chapter, is a well-known and researched field. One could argue that a reasonable choice for the selected *off-the-shelf* object estimator method employed in the presented framework would be a system engineered to accommodate objects which are grasped or in the vicinity of hands. Bearing this in mind, we claim these lack some years of usage by the research community to supersede the standard methods. Furthermore, wanting to investigate the robustness of our pipeline to the presence of unknown object instances, we decide to choose a category-level pose estimator. The latest category-level method we could find that made available their source code was CAtategory-level Pose Tracking for Rigid and Articulated objects (CAPTRA) [1], thus being this the method we use.

Architecture

CAtategory-level Pose Tracking for Rigid and Articulated objects (CAPTRA) addresses 3D translation, 3D rotation, and 3D size estimation of rigid or articulated objects from known categories, establishing 9DoF object pose estimation and tracking. Weng et al [1] present an end-to-end differentiable system that jointly canonicalizes input and output spaces. Besides removing the need for any pre or post-processing, this formulation provides an accurately fast estimation and tracking of object poses.

As a tracking method, CAPTRA improves on the previous estimation. A good initialization is then a must for proper usage of the system. Such limitation had an impact on the formulation

of the task elaborated and tested using our framework. However, this restriction is acceptable compared to the stability improvement tracking methods offer for tasks where visual servoing smoothness is foreseen, as in-hand manipulation. A Point Cloud (PC), along with the prior pose estimation, is the input for the current examination. The output pose estimation will then accompany the succeeding Point Cloud data as the inputs for the next iteration, closing the loop.

This project’s focus is the in-hand manipulation of a rigid object, a water bottle. Therefore, the following explanation of the PC path through CAPTRA’s networks will comprise merely the rigid object formulation.

Pose Canonicalization

For a particular time instant, we have point cloud $X = \{x_i \in \mathbb{R}^3\}_{i=1}^N$, containing rigid object instance C . The category-level pose is $P = \{d, R, T\}$, where $d \in \mathbb{R}^3$ is 3D size, $R \in SO(3)$ is rotation, and $T \in \mathbb{R}^3$ is translation. P , similar to Wang et al [27], is factorized to a 7 DoF similarity transformation $\mathcal{T} = \{s, R, T\} \in Sim(3)$, with $s = \|d\|$ being a 1D uniform scale.

\mathcal{T}_t and X_{t+1} combine according to

$$Z_{t+1} = (R_t)^{-1}(X_{t+1} - T_t)/s_t, \quad (3.1)$$

originating the canonicalized Point Cloud Z_{t+1} . Analogous to X containing C , Z_{t+1} contains \hat{C}_{t+1} .

$$\hat{C}_{t+1} = (R_t)^{-1}(C_{t+1} - T_t)/s_t, \subset Z_{t+1}. \quad (3.2)$$

The method then maps, in Normalized Object Coordinate Space, Z_{t+1} to $\hat{\mathcal{T}}_{t+1}$. \mathcal{T}_{t+1} , can then be expressed by uniting $\hat{\mathcal{T}}_{t+1}$ and \mathcal{T}_t , as

$$\begin{aligned} s_{t+1} &= s_t \hat{s}_t, \\ R_{t+1} &= R_t \hat{R}_t, \\ T_{t+1} &= s_t R_t \hat{T}_t + T_t. \end{aligned}$$

The usage of NOCS allows better generalization across instances and improved accuracy in the differentiable regression task.

Pose Tracking

As shown in figure 3.2, CAPTRA’s core is composed of two Point Cloud Neural Network (PCNN), CoordinateNet, and RotationNet, each processing part of the canonicalized PC Z_{t+1} . The non-linearities of $SO(3)$ can be bypassed by NOCS, where the \hat{R}_{t+1} can be directly and ac-

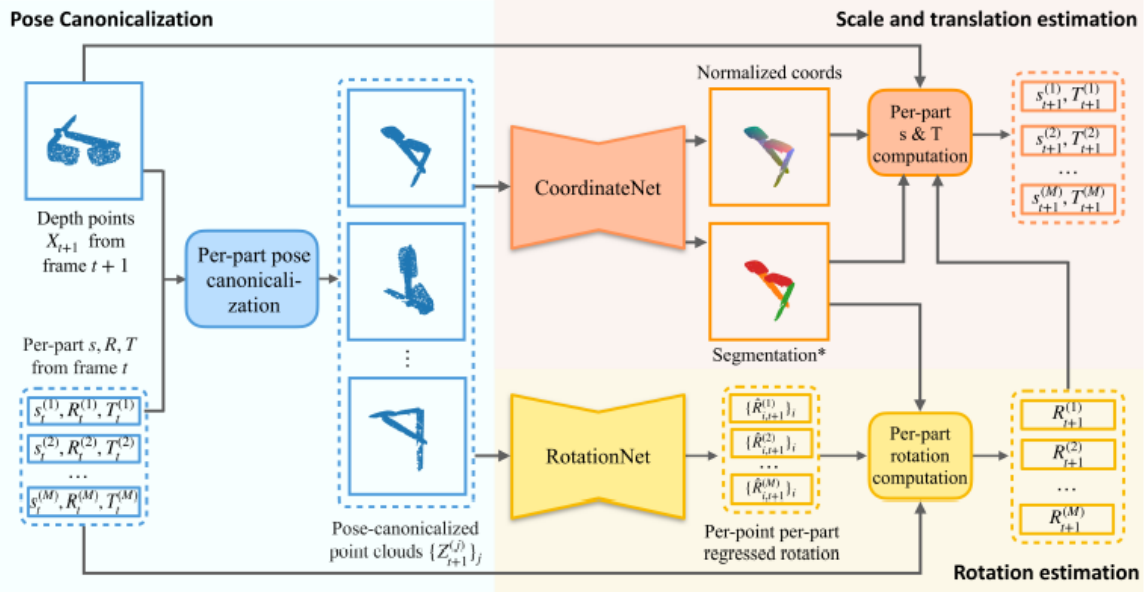


Figure 3.2: CAPTRA differentiable pose tracking pipeline [1]

curately regressed by the RotationNet since minimal variations exist with respect to the identity matrix $I_{3 \times 3}$.

Despite holding the same properties, occlusion situations make difficult the prediction of the translation and size. Selecting a bottle as an example, since it is the item we use in our experiments, one can agree that estimating its dimensions and center is troublesome if any of the tips is occluded. One alternative is to compute normalized coordinates Y_{t+1} that catch the category-wise priors and implement a zero-centered frame, which implicitly predicts the \hat{s}_t and \hat{T}_t in a more reliable way than directly regressing them. Joining Y_{t+1} computed by CoordinateNet with the R_{t+1} of RotationNet, as in

$$C_{t+1} = s_{t+1}R_{t+1}Y_{t+1} + T_{t+1}, \quad (3.3)$$

the s_{t+1} and T_{t+1} can later be predicted using the Umeyana algorithm, whose usually requirement for non-differentiable post-processing on Y_t , as RANSAC, is circumvented by CAPTRA's pose canonicalization. Faster and more precise estimations, as well as an optimization on pose losses, is this way possible to CAPTRA's end-to-end differentiable PCNNs.

Weng et al [1] did not design their PCNNs from scratch, basing RotationNet on PointNet++ and growing CoordinateNet on top of the segmentation network of PointNet++. However, PointNet++ is not deterministic due to random further point sampling in a couple of operations, creating convergence challenges. RotationNet regresses on a per-point basis, in the form

of 6D continuous rotation representations, and completes with Euclidean mean to mitigate noise negative effects. A key aspect is the ability of RotationNet to deal with issues symmetric objects create, like the ones the bottle used in our experiments poses. The Point Cloud Neural Network tackles this issue by regressing the 3D end-point position of the bottle's unit symmetry axis, leading the redundancies to amiable and continuous regression rotation representations.

The symmetry challenges with the rotation propagate back to the normalized coordinates computed by CoordinateNet. CoordinateNet, backbone by PointNet++ segmentation network, has two branches after the final propagation layers, one for segmentation and another for prediction. The segmentation branch operates with a relaxed version of Intersection over Union (IoU) loss while the prediction branch estimates class-aware normalized coordinates. Since point-pairwise distances are invariant to rotations, Euclidean and symmetric coordinate loss are coupled to help supervise the predictions for symmetric object instances.

Results

Former *state-of-the-art* in category-level object pose estimation for rigid instances, Pose Anchor-based Category-level Keypoint tracker (6PACK) [2], is highly surpassed by CAtegory-level Pose Tracking for Rigid and Articulated objects (CAPTRA) [1]. According to the authors, an absolute increase of 40% in mean accuracy and 10% in mean IoU metric is reached.

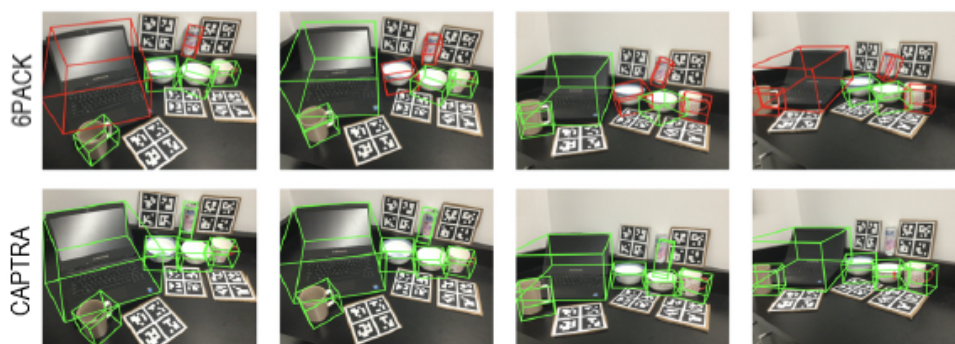


Figure 3.3: Result comparison - 6PACK [2](above) and CAPTRA [1] (below) - retrieved from [1]

NOCS dataset contains six categories of items - bottle, bowl, camera, can, laptop, and mug. Half of these, the bottles, bowls, and can, are symmetric instances. For the pose tracking of rigid objects, CAPTRA registers four different metrics:

- the percentage of pose predictions where both rotation and translation errors are below 5° and 5cm respectively - 5° 5cm accuracy(%)

- the average 3D IoU between the predicted bounding boxes and the ground-truth information - $mIoU(\%)$
- the average rotation error - $R_{err}(\circ)$
- the average translation error - $T_{err}(\text{cm})$

These metrics results, averaged over all the six NOCS categories, are reported in table 3.1, comparing CAPTRA under RGBD or depth only implementations and 6PACK RGBD implementation. The initialization of the methods for these experiments encompasses pose perturbations from the mask segmentation, with one of 6PACK’s results concerning only translation perturbation.

Table 3.1: Pose tracking results averaged over all the six NOCS categories - retrieved from [1]

Method	NOCS [27]	6PACK [2]		CAPTRA [1]	
Input format	RGB-D	RGB-D		Depth	RGB-D
Initial perturbation	-	Translation	Pose	Pose	
5° 5cm accuracy($\%$) \uparrow	16.97	28.92	22.13	62.16	63.60
$mIoU(\%)$ \uparrow	55.15	55.42	53.58	64.10	69.19
$R_{err}(\circ)$ \downarrow	20.18	19.33	19.66	5.94	6.43
$T_{err}(\text{cm})$ \downarrow	4.85	3.31	3.62	7.92	4.18

6PACK was used in real robot tracking experiments, obtaining reasonable accuracy and consistency for the bottle instances. Moreover, the robot also accomplished manipulation tasks, pouring or tossing, based on the pose of the bottle that tracked. Combining the documented superiority of CAPTRA over 6PACK with the robot experiences described by 6PACK’s authors, and taking into account the tracking speed results of table 3.2, we were confident that using CAPTRA as the visual estimator in the first part of our pipeline had good success probabilities.

Table 3.2: Comparison of tracking speeds in frames per second (FPS) - retrieved from [1]

Method	NOCS [27]	6PACK [2]	CAPTRA [1]
Tracking speed (FPS)	4.05	3.53	12.66

3.2.2 Model-free manipulation learning algorithm - DAPG

The second part of the proposed framework is the manipulation control mechanism. For this, we resorted to trending tools in the field, learning ones in particular, which have been highly

used in similar tasks for the last couple of years. With the hardware available for this project, from the end-effector to the computational power, our research was guided to less ambitious yet relevant tasks.

Manipulation control algorithms, as detailed in the previous chapter, have seen different protocols over the years. Keeping up with the times, we resort to another Neural Network to accommodate the selected manipulation method. One could argue that the selected task, described in detail in the next chapter, is too simple for the use of such a powerful and expensive algorithm. Bearing this in mind, we chose to disregard such argumentation with the intent of exploring real environment possibilities of the Reinforcement Learning algorithms, which are commonly just encountered in simulation scenarios. The work of Zhu et al [15], with promising results with the Demo Augmented Policy Gradient (DAPG) algorithm, led us to consider the same algorithm for our pipeline.

Notation

The control part of our pipeline is modeled as an Markov Decision Process tuple \mathcal{M} , introduced in section 2.1.2, 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 π is denoted D^π . The standard definitions of value, Q, and advantage functions are given by

$$\begin{aligned} V^\pi(s) &= \mathbb{E}_{\pi, \mathcal{M}} \left[\sum_{t=0}^{\infty} \gamma^t r_t \mid s_0 = s \right], \\ Q^\pi(s, a) &= \mathbb{E}_{\mathcal{M}} [\mathcal{R}(s, a)] + \mathbb{E}_{s' \sim \mathcal{T}(s, a)} [V^\pi(s')], \\ A^\pi(s, a) &= Q^\pi(s, a) - V^\pi(s). \end{aligned}$$

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 π 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 π .

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 $\eta(\pi) = \mathbb{E}_{\pi, \mathcal{M}} [\sum_{t=0}^{\infty} \gamma^t \mathcal{R}_t]$.

DAPG belongs to the aforementioned policy gradient optimization methods. In order to make the algorithm more robust, instead of 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 Behavior Cloning, formulated by

$$\underset{\theta}{\text{maximize}} \sum_{(s,a) \in D} \ln \pi_{\theta}(a|s). \quad (3.4)$$

As concluded experimentally by [3] and [15], 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 Natural Policy Gradient (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

$$g = \frac{1}{NT} \sum_{i=1}^N \sum_{t=1}^T \nabla_{\theta} \log \pi_{\theta}(a_t^i | s_t^i) A^{\pi}(s_t^i, a_t^i, t), \quad (3.5)$$

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. More details on these computations, we recommend the reader to take a look into [81].

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

computed as

$$F_\theta = \frac{1}{NT} \sum_{i=1}^N \nabla_\theta \log \pi_\theta(a_t^i | s_t^i) \nabla_\theta \log \pi_\theta(a_t^i | s_t^i)^T. \quad (3.6)$$

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

$$\theta_{k+1} = \theta_k + \sqrt{\frac{\delta}{g^T F_{\theta_k}^{-1} g}} F_{\theta_k}^{-1} g, \quad (3.7)$$

where δ is the chosen step size.

REINFORCE's gradient, in equation 3.5, is augmented to

$$g_{aug} = \sum_{(s,a) \in D^\pi} \nabla_\theta \ln \pi_\theta(a|s) A^\pi(s,a) + \sum_{(s,a) \in D} \nabla_\theta \ln \pi_\theta(a|s) w(s,a), \quad (3.8)$$

where the weighting function w

$$w = \lambda_0 \lambda_1^k \max_{(s',a') \in D^\pi} A^\pi(s',a') \quad \forall (s,a) \in D, \quad (3.9)$$

combines the relevance of the demonstrations towards the policy decisions, with k being the iteration counter.

The additional regularization term in (3.8) 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 [3] 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 $\lambda_1 = 0.95$.

Results

Manipulation tasks suffer from several discontinuities and constraints imposed by joint limits and frequent contact changes, making accurate gradient estimation methods unsuitable. Rajeswaran et al [3] designed an RL-aided IL algorithm to circumvent these problems, verifying their work on four manipulation tasks in a simulated environment. One of the four experiments is in-hand manipulation of a pen by a 24 DoF ADROIT hand. 25 demonstrations, collected through a computational expert due to difficulties with real scenario data gathering, equipped

DAPG’s Imitation Learning initialization.

Table 3.3: Pen in-hand manipulation task - Sample and robot time complexity - retrieved from [3]

Method	DAPG [3]	NPG	
Reward	Sparse	Shaped	Sparse
N	30	864	2900
H	3.33	96	322

The huge improvement of DAPG over NPG with shaped or sparse completion reward is visible in table 3.3. Here N is the number of iterations, constituted of 200 trajectories of 2 seconds each, required to possess a 90% success rate, and H is the robot hours needed to do so.

DAPG’s simulation results on such complex tasks propelled its trial in real environment experiments, where Zhu et al [15] work would come to confirm the algorithm’s potential. For the real-world scenario, Zhu et al [15] recreated three tasks inspired by everyday hand activities, two of which are the valve rotation and the vertical box flipping. The finger of the end-effectors must learn to cooperate, simultaneously pushing and getting out of the way, in order to rotate the faucet to a target position or to turn the vertical box that freely spins about its horizontal axis. 20 expert trajectories, collected via kinesthetic teaching, guided the learning of coordinate movement patterns of two different manipulators, the Dynamixel claw and the Allego hand.

Table 3.4: Real environment tasks training times in hours - retrieved from [3]

Method	DAPG [3]	NPG
Valve rotation (hrs)	3.0	7.4
Box flipping (hrs)	1.5	4.1

Table 3.4 highlights the superiority of DAPG over Reinforcement Learning without IL, this time in a real-world scenario. It can be observed that the training time in hours required to achieve a 100% success rate over 10 evaluation rollouts is reduced by a factor of 2.

Zhu et al [15] provided reward ablation studies that allowed us to be inspired by their formulation for our own experiments. The following three rewards presented results

$$\begin{aligned}
 r_1 &= -\|\theta - \theta_{goal}\|_2, \\
 r_2 &= r_1 + 10 \cdot \mathbb{1}_{\{r_1 < 0.1\}} + 50 \cdot \mathbb{1}_{\{r_1 < 0.05\}}, \\
 r_3 &= r_2 - \|v\|_2.
 \end{aligned}$$

The first two reward forms were considerably better than the third, despite the latter possessing

a control regularization term for smoother movements. As a matter of fact, such regularization resulted in a poorer exploration part and consequently to more unsatisfactory results. As the time steps increase, the first two reward formulations present a similar decay, approaching zero mean distance to the goal, while the third stagnates and seems not to converge in a way to cancel the error to the objective.

Opting by the second reward function for both tasks described above, Zhu et al [15] concluded that the absence of the control regularization term did not cause unsafe manners in the trained policies. Not wanting any accident in the laboratory during the experiments conducted for the development of this thesis, such guarantees led us to opt for a similar reward form, as will be described in the next chapter. We can vouch for the reward selection as no person was harmed nor any equipment damaged in our experiments.

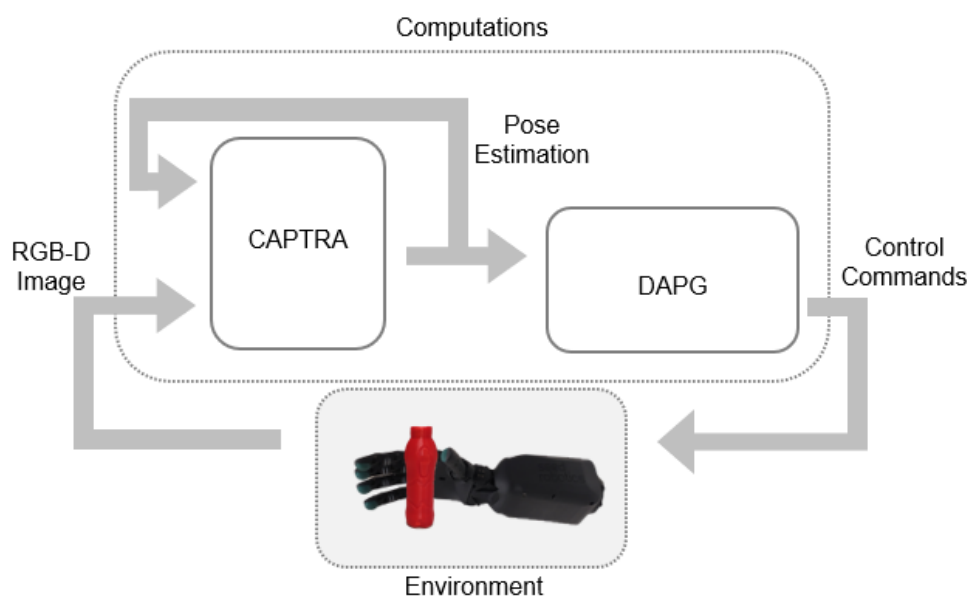


Figure 3.4: Proposed framework - coupling of CAPTRA [1] with DAPG [3]

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 the methods detailed in the previous section to form our framework, as depicted in figure 3.4. The proposed diagram implements the general one introduced in the previous section for the particular task experimented with within this thesis. Our project distinguishes amongst other similar works, as Zhu et al [15], by the adaptation of a learning scheme Rajeswaran et al [3] to encompass non-ideal estimation information.

Both CAPTRA's and DAPG's had, by the date of writing of this project, a python version of

their code implementation available on GitHub, which we adapt to our experiments. Two possibilities for merging these methods were in a sequential or parallel fashion. The first performs a pose estimation between each hand movement, providing the needed information to the control method. The latter continuously estimates the pose at the fastest rate possible, and the control part of the system queries for information when it is required.

The two alternatives have pros and cons, and after trying on the empirical setup with both options, the final tests were conducted in a sequential manner. Sequential code implementations are easier to elaborate and less demanding on computer hardware specifications. However, the time bottleneck of the experiments was not the computer computational power but in the collection of on-policy data, as expected. Besides, Zhu et al [15] had already highlighted that hand-object interactions in a real scenario are costly, thus shifting the decisions towards the more straightforward method for object pose prediction.

In theory, as noted in section 3.2.1 as well, the pose tracker would work best if operated in parallel with the bottle being manipulated in a way that allowed it to capture the bottle's pose faster than its movement. The first experiments were executed in such a fashion bearing this in mind, later understanding that the available FPS could not deal with the actual problematic situation, the contact forces on the bottle.

Abrupt movements of the fingers lead the bottle to comply instantly with the rules of physics, which might cause it to shift its position beyond the range expected by the pose estimation tracker, resulting in a loss of the track of the object. Containing these, as we did by constraining the fingers' position discrepancies between consecutive commands, mitigates the problematic situations, even though not completely eliminating them, as gravity could still cause a few tracking hurdles. Analyzing this situation, the advantages of parallel estimation and control were no longer visible, which took us to the sequential and more reliable implementation.

3.3 CAPTRA adaptations

The *state-of-the-art* achievements reported in academic papers gain more recognition within the community when it is possible to replicate its results, often leading to the disclosure of the source code on platforms like GitHub. In the object detection and estimation field, researchers tend to divulge code particularly suited for a dataset, despite any further applicabilities of the algorithms present in the research article. Weng et al [1] argues on the simplicity of operating CAPTRA with live stream data. However, the Python implementation, available at the beginning of the development of this project, was bound to two different datasets, one for the

articulated object instances, the other for the rigid object instances, the NOCS [27] dataset.

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 illustrated in figure 3.2, in section 3.2.1. We noticed that the object segmentation part, a requirement for the live stream implementation, was not disclosed in CAPTRA's code since it was coming directly from the NOCS [27] dataset pairs. To solve this, we resorted to the simplest form of object segmentation, the color. Further details are provided ahead in this section.

After cleaning the code to salvage the skeleton PCNN's, we too had to adapt it to our needs, that is, changing experimentally dependent values such as the camera's intrinsics used in our experiments to capture the RGB-D image of our object, the bottle. We selected the bottle from the six categories NOCS was pre-trained on since we thought it could yield the best results for the in-hand manipulation task, and training of a Deep Neural Network is beyond the scope of this thesis.

3.3.1 Segmentation methodology

Migrating to a live stream implementation of CAPTRA required a segmentation mechanism not present in the available Python 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.

The hardware peculiarities are visible in the experimental setup documented in the next section of this chapter. We consider that, for the controlled scenario where the experiment was conducted, the choice of the color segmentation algorithm presented the best relation between accuracy and time and monetary requirements. The attention shifted then towards particular details of the method as to which was the most appropriate color space.

RGB color space is assumably the most known, being present in a panoply of everyday home appliances, from the television to the screens and cameras of our smartphones. However, numerous spaces exist, each with its own strength. The task of color segmentation possesses a color model with distinct superiority, the Hue Saturation Value (HSV), leading the first trials for this feature in our particular CAPTRA implementation to make use of HSV's advantages.

We were already anticipating problems regarding the finger occlusions pertained to the in-hand manipulation of the bottle, so initially, we tried to have a fine-grain color filter. Such a task,

possible yet difficult to obtain even with an HSV color scheme, was the best option available to obtain better results for the object's pose.

As a matter of fact, trying with an effortless RGB gamma decayed filter led to fewer hand-wrought needs and better stability, even with the mask catching points away from the bottle, like the forearm screws of the manipulator. The gamma decayed within the RGB filter was the solution encountered to fight the camera's non-adjustable white balance by forcing a darkening of the received image. The final experiments relied on this RGB gamma decayed filter, with the gamma value selected empirically in a way to diminish the light influence within the estimation to a minimum.

The interface used to communicate between the cameras acquiring the RGB-D images and the main computer running our CAPTRA + DAPG framework is Robot Operating System (ROS). The integration of different program or code versions continues to be a huge problem for every engineer that wants to use the work of others to advance in its research. It was no different in this thesis. After harsh endeavors, we managed to mix hardware and software for the object pose estimation, with OpenCV's Bridge package living up to its name and connecting the images collected from the cameras and the Python function processing them.

3.4 DAPG adaptations

The experimental work present in this thesis relates to Reinforcement Learning aided by Imitation Learning based on the selection of DAPG as the control algorithm for our manipulator. As one can expect, Rajeswaran et al [3] article details a hybrid IL-RL workflow executed in a simulation environment, with further inspection being possible since the Python code of their algorithm is present on GitHub. Despite Zhu et al [15] investigation being hand-to-hand with ours, their real-world DAPG implementation was not publicly available at the time of writing of this thesis, pulling our starting point to the original and disclosed DAPG version.

Our proposed framework leans on the core structure of Rajeswaran et al [3] work, as likely Zhu et al [15] do too. As noted in the previous chapter, Zhu et al [15] transpose DAPG's algorithm from simulation to the real world by removing Mujoco's physics simulator dependencies. We then modified the same 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 new Python class. Analogous to the OpenCV Bridge package, our Python class is responsible for bridging the hardware and software at the laboratory by managing the interactions between the

setup and the computations.

The close resemblance between Rajeswaran et al [3] and our control component extends from the training format to the Deep Neural Network parameters, which were kept equal to the originals for the final experiments. Table 3.5 illustrates such values, where the iteration number presented corresponds to the ideal for full training the network. As far as we know, Zhu et al [15] results were achieved with the same DAPG hyperparameter values, apart from the initial variance of the policy for exploration that was a bit tuned. We had then strong reasons to believe these values were going to allow us fast learning of the intended task. Further research may investigate other parameter values more suitable for real-world usage of this RL-aided IL algorithm.

Reward, states and actions

In section 3.2.2, we introduced that our reward function resembles one of the three tested by [15], which in turn had already a strong Rajeswaran et al [3] basis. The reward function employed in our experiments is then described by

$$r = -\|\theta - \theta_{goal}\|_2 + 10 \cdot \mathbb{K}_{\{\|\theta - \theta_{goal}\|_2 < 75^\circ\}} + 50 \cdot \mathbb{K}_{\{\|\theta - \theta_{goal}\|_2 < 60^\circ\}} \quad (3.10)$$

where the selected values were empirically proposed based on an analysis of the demonstration data collected and the difficulty of the proposed task. It is important to highlight that such a reward shape is only possible due to an approximation we made regarding the available Degree of Freedom 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 4.1, the angle between the bottle’s revolution axes and the horizontal is obtained from a projection of CAPTRA’s object pose estimation.

The reach of the taken approximation expands beyond the reward function into the state space formulation. Once again, we took inspiration from Zhu et al [15] to build the state representation that our neural network, responsible for the control mechanism of the hand, receives. The dual hidden layer multilayer perceptron (MLP), containing 32 neurons per layer, takes as input the environment observations, encoded in the state representation, and provides control commands for the robotic hand manipulator, in the form of motor angles, as output.

With the training loop constituting a recurrence of action realization, action observation, and repeat, we believe adequate to incorporate the actions taken in previous time steps into the

Table 3.5: Training parameters

Parameters		
	Network	[32, 32]
B	learning rate	0.001
C	batch size	32
D	learning rate	0.001
	batch size	64
A	step size	0.05
	gamma	0.995
P	lambda 0	0.01
	lambda 1	0.95
G	Number of iterations	100
	Trajectories per iteration	200

state representation received by the learning network. The environment state we considered in our experimental task, documented in more detail in the next chapter, comprises $(5+5)+(1+1)$ values. These 12 values, expressed in a non-orthodox manner to easily highlight their grouping, 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 introduced in section 2.1.2. 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. One could argue that, before such approximation, the use of a category-level pose estimator may be excessive. Aware of this, we argue that the purpose of the proposed framework is to accomplish further tasks than the one described and investigated in this thesis. The final value of the state representation is the angle from the current inclination of the object to the goal angle, the desired 45° . This computation is also present as the first term of the reward function depicted above.

4 | Experiments

Visual and tactile stimuli are the most indispensable feedbacks a person possesses when it comes to object manipulation. Coveting for robots' intelligence and dexterity to rival humans' wisdom and handling skill, a natural and required unity of vision methods and learning algorithms takes place.

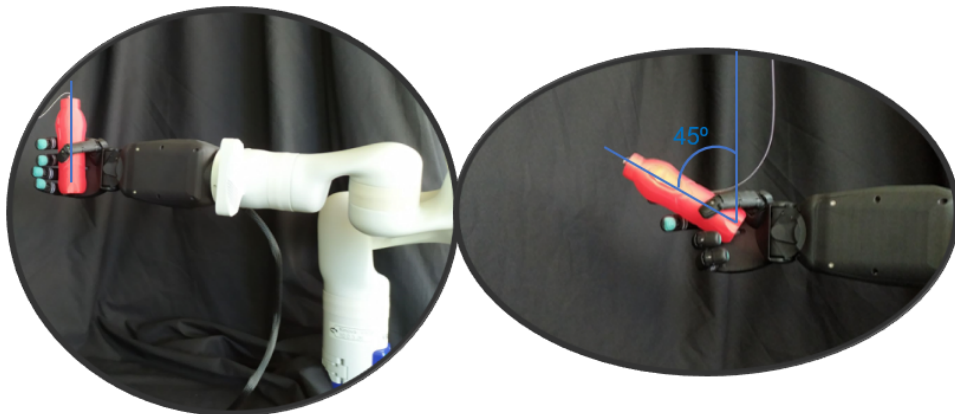


Figure 4.1: In-hand manipulation task explored in this thesis - initial state (left) and goal state (right)

Figure 4.1 illustrates, 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 thesis. The final pose of the bottle should reach a 45° 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° of the goal state for 5% of the trajectory time. We do not bound this percentage to be in the final moments of the trajectory *per se*, as the task's physics laws already enforce, in a way, such limitations.

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.

This chapter provides a comprehensive report of our experimental procedures, from data gathering processes to results analyses and discussion, not forgetting the environment setup.

4.1 Equipment and set-up

Experimental set-up

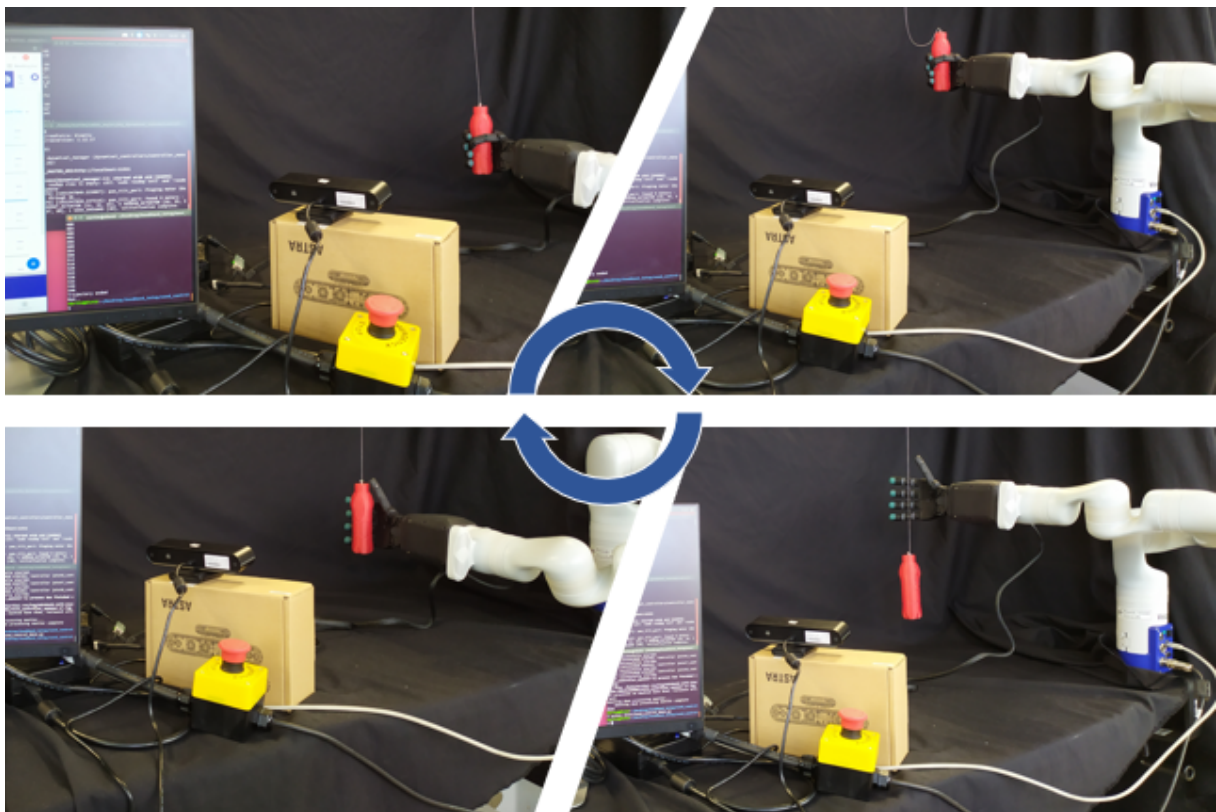


Figure 4.2: Experimental environment - The learning cycle comprising: grasping (upper left) the bottle, lifting it up(upper right), executing in-hand movements to tilt the bottle that will eventually drop it (lower right), and re-positioning the hand in a way to start the cycle all over again (lower left)

The figure above illustrates the different stages of the learning cycle with actual pictures of the laboratory setup we used during this thesis. We can observe the experimental scenario, which comprises the essential elements for the job described in this document

- RGB-D camera for object pose estimation

- Humanoid end-effector to manipulate the object
- Handled object
- Computer to store and compute the required estimation and actuation parameters

Specifications of each one of these components, except for the computer, will be given next. Not fundamental material, yet highly encouraged to have in a real-world Reinforcement Learning experiment, is a reset procedure. 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 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.

Behind the scenes

Reinforcement Learning experiments in a real scenario are intensively time-consuming. Previous DAPG works [3, 15] estimated the learning time in a few hours for the tasks. Preliminary trials with our assembled setup show different time estimations. Running a learning trajectory, defined as the action-state pairs that occur from the initial grasp until either a goal state is reached or the object falls from the manipulator, takes on average 2/3 minutes to complete. This value is a rough empirically estimation based on the above-mentioned preliminary trials, as the learning trajectories, in theory, tend to grow with time since the robot starts to understand how to avoid dropping the bottle on the first moves.

Taking a look at table 3.5 in section 3.4, which synthesizes DAPG's training parameters used by Rajeswaran et al [3], we can make an educated guess that each of the one hundred iterations, comprising two hundred learning runs, could take around ten hours to complete. With the reset mechanism implemented, which theoretically should suffice for an autonomous, non-stop execution of the Reinforcement Learning trials, the training could be completed in forty two days. In practice, one must remember that mechanical machines overheat if used continuously. So, every three to four hours, we had to let the material cool down to avoid jeopardizing the hardware. Besides this fact, our experiments required constant supervision as the string affixing the 3D printed bottle to the ceiling could get entangled with the fingers of the robotic manipulator. These situations, depicted below in figure 4.7, may damage the end-effector if not quickly resolved.

Summing all this, we are looking at a really long time to conclude real-world training in the same manner Rajeswaran et al [3] did in simulation. An experiment with a long life span, requiring several university materials, may have an impact on investigations conducted by other researchers. Moreover, for security reasons, the university's policy is to store the most expensive hardware, such as the robotic arm, in the respective suitcases when not being used. Therefore, to permit the storing and the sharing of materials without conducting an expensive re-calibration process every time we stop the experimentations, a robust assemble/disassemble strategy must be part of our experimental setup.



Figure 4.3: Experimental environment - structure

The support structure of the laboratory-created environment is crucial to, besides supporting the black blanket for CAPTRA's color segmentation, restrict all elements to their positions. In figure 4.3, we can observe the plywood skeleton that sustains the black background. Together with this, the visible small plywood markers adhered to the table facilitate the setting of the robotic arm base and the RGB-D camera into their position. Appendix A specifies in more

detail how to recreate this experimental environment if one wants to extend our work in future research.

Side by side with the re-position of the hardware, the software required some adjustments to permit stopping and re-starting of the training Reinforcement Learning runs. The DAPG author’s Python released code already contemplates a saving function of the learned weights for the manipulation model at the end of each iteration. We added learning trajectory backups every five runs to accommodate for the time and mechanical constraints.

Camera



Figure 4.4: RGB-D camera position markers

The relative position of the camera with respect to the robotic hand must be constant throughout the experiments to provide valid pose predictions. In combination with the plywood table’s clung marker that indicates the camera’s box position on the desk, visible above in figure 4.4, laboratory tags were stamped into the camera’s package to define the orientation of the camera during the experiments.

We used one of the Orbbec Astra 3D cameras available in the laboratory during our experiments - figure 4.5. As with many other peripherals, plug-and-use drivers are available for this camera, leading to a forthright acquisition of RGB-D images for CAPTRA’s pose estimation. As stated above in section 3.3, CAPTRA’s released code implementation is particularly adapted for the NOCS [27] dataset, from expecting mask images as input to using NOCS camera intrinsics and object sizes in the pose computations. In order to shift from Normalized Object



Figure 4.5: Orbbec Astra RGB-D camera

Coordinate Space to live stream usage of CAPTRA, we created the supplementary color segmentation described under section 3.3.1 to supply the needed mask matrices. Alongside these, the necessary depth matrices, retrieved from the RGB-D images, required a change of NOCS's camera intrinsics to Astra's intrinsic parameters.

While having a contrasting background is key for the developed color segmentation technique to work properly, stable light conditions also severely impact the quality of the retrieved mask matrix. The available space in the laboratory for the assembly of our experimental setup happened to be near a window, which provides good natural illumination. Wanting the possibility to collect data and train our robotic hand at any given time, and knowing that natural illumination is not stable enough for the task at hand, we always maintained the headlamps on, whether night or day, rain or shine.

Hand

An underactuated Seed Robotics RH8D humanoid hand was the manipulator of choice for the executed experiments. This 7 Degree of Freedom end-effector possesses seven controllable motors. The wrist and the thumb contain two each, while the index and middle fingers get one apiece. The last motor manages the ring finger and the pinky jointly. As communicated above,



Figure 4.6: Seed Robotic RH8D hand

to respect the in-hand manipulation premise, we freeze the wrist joints to a natural rest position, leading to only five out of the seven motors being part of our action space. Each motor, driven by a radian angle between 0 and 2π , will pull the tendon wire of its associated finger in order to bend it. The ring finger and the pinky will bend as one, while the thumb, besides flexing, also revolves. These radian values are transferred from our developed Python class, mentioned in section 3.4, through a ROS script.

The Seed manipulator is entirely, apart from the fingertips, made of non-deformable flat plastic. Despite the Seed fingertips containing tactile sensors, as depicted in figure 4.6, we did not use this tactile feedback in our experiments as a way to simplify the initial approach to the in-hand manipulation task. From the start, one can observe that such a rigid structure will impose severe difficulties for any in-hand manipulation task since this hand lacks the capability to do some critical maneuvers a hollow human palm can. Furthermore, the underactuated situation fatally restricts the abilities a finger on its own can achieve.

On the other hand, such a robust design brings along elastic joints, as shown in figure 4.7. Here, a safely recreating of an experienced failure situation during the Reinforcement Learning trials is illustrated. The string got intertwined into the finger, and the reset procedure caused the finger to twist in a weird and harmful way. Thanks to the constant supervision over the

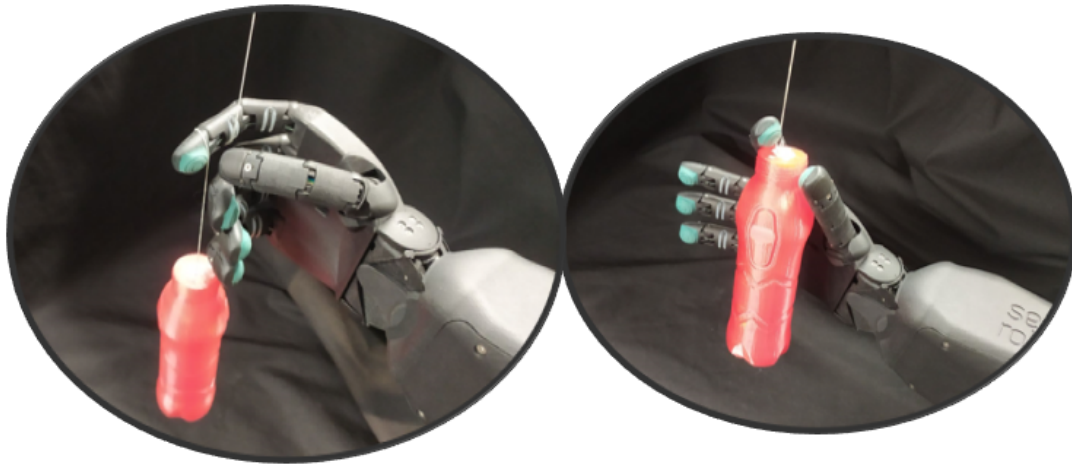


Figure 4.7: Seed Robotic - finger nearly rupture situations

experimental procedure, whenever one of these rare malfunction situations occurred, we were able to resolve them instantly, without damage to the equipment or the collected data.

Arm

The training of a Neural Network requires huge amounts of data. Acquiring training data through a trial and error approach is thus intensively time-consuming. Without a reset mechanism, our experimental in-hand manipulation task would be unbearable. Thanks to the robotic arm displayed in figure 4.8, we could rely on the effortless reset solution described earlier.

Gravitational forces plus the hooking up the 3D printed bottle to the ceiling via an invisible string ensure that, when dropped by the Seed hand, the bottle will, after an initial transitory swing motion, always end up in a known and constant position in the 3D space. We can denominate this location as the reset position, and the area where the Seed manipulator executes the RL trial runs as the train position. During the Reinforcement Learning training procedure, the arm's job is to relocate the hand between both locations when solicited by the software.

We used a Kinova Gen 3 robotic arm with 7 Degree of Freedom. Kinova's arm has the possibility to attach a particular type of end-effectors and control them through the same API that operates the robotic arm, which unfortunately is not the case with our humanoid hand. Moreover, Kinova's junction terminal does not respect the ISO 9409-1-50 norm, the universal standard adapter common in most robotic arms. Nevertheless, linking the Seed Robotic with the Kinova was doable by 3D printing a mounting adapter piece, seen in figure 4.9.

With Kinova's API, we can command the arm to go to any reachable 3D location in the 3D space, either by supplying the cartesian coordinates with respect to Kinova's base frame or by



Figure 4.8: Kinova Gen3 arm

providing a vector with the seven joint values for the arm. The API further allows combining successive of these simple blocks to create trajectories. Our project's reset operation mandates the creation of two trajectories, namely a reset one, from the train position to the reset position, and a lift one, from the reset position to the train position.

Apart from these, a path between the arm's packing position and the initial reset position is also needed, not forgetting a stop along the way to couple the Seed manipulator to the robotic arm since they do not pack jointly. Kinova arm's ability to be freely operated by a user, in a special acquisition mode which also allows recording particular joint values in the midway, enabled a smooth and quick creation of our custom trajectories.

The Kinova arm available in the laboratory is equipped with an Intel RealSense depth sensor near the end-effector. If severe occlusion was not a critical issue in the object's pose estimation, this could have been the only camera used in our experiments. As this is not the case, we utilized the external Astra RGB-D camera, precisely positioned in order to have a better field of view of the bottle, to perform the object's pose prediction. The drawback of such Astra positioning was the difficulties in detecting if the bottle had slipped the control of the Seed hand. For this vital



Figure 4.9: Adapter to couple the Seed Robotics hand to Kinova Gen3 arm

role, we resorted to the Kinova’s Intel camera. We apply the color segmentation mask, catching the fall of the bottle and commanding the robotic arm to take the Seed hand to the reset position.

Bottle

CAPTRA made available six pre-trained NNs weights corresponding to the six rigid objects categories of the NOCS dataset [27]. For our experiments, since the training of a Neural Network is beyond the scope of this thesis, we selected the item that seemed most promising among the available categories, comprising bottles, bowls, cameras, cans, laptops, and mugs. After some quick experiments with a 33cl soda can and a 33cl water bottle, we opted for the water bottle for its smaller diameter size, which was more appropriate for the Seed Robotic hand.

Referring back to section 3.3.1, we needed to extend CAPTRA’s open code implementation with a segmentation method. A straightforward color segmentation technique was way more effective on a magenta soda can than on a translucent water bottle. Luckily, we managed to use an available 3D printer to make a customized 3D printed opaque water bottle. In fact, we happened to create 3 in the course of the experiments, as can be seen in figure 4.10.



Figure 4.10: Bottle's 3D printing iterations

The already detailed experimental setup, also illustrated in figure 4.3, has a black blanket as the background, supported by the white plywood structure. The black-white contrast is the highest possible. Since both the robotic hand and the background blanket are black, the natural choice for the first item is a 3D white opaque bottle. According to the manufacturer, the weight of the items manipulated by the Seed Robotic manipulator can not exceed 750g, which led us to print an empty bottle, ending up with a light opaque 3D printed white hollow bottle to manipulate.

The hollow object turned out to be too weak to hold the pressure imposed by the manipulator's fingers, leading to a second iteration of the bottle. The white 3D filament was not available, so we went for a green one without realizing this was the color of the fingertips of the Seed hand. We were confident, at this stage, that a clean and precise mask of the object carried the best chance of succeeding in the manipulation task, so the selection made was far from ideal. Further mistakes were made concerning the filling of the bottle, where we, trying to avoid a

copy of the deformable white water bottle, filled around 75%, which created a cement block. A heavy opaque 3D printed solid green bottle was then the second, not yet final, iteration.

The object on the left in figure 4.10 is the final version of the bottle, the one used by the DAPG learning algorithm. A shining red was the selected 3D filament. Despite not being as contrasting with the background as white, red is not a color present in the manipulator or any other material within Astra's camera field of view. The filling was reduced to around 45%, nearly half of the second bottle version, keeping the structure, including the lid, fortified but reducing unnecessary weight. The bottle's lid is part of the learning structure, so to say, as it will support the bottle's falls during the learning process. Notwithstanding this effort, during the course of the experiments, a bit of hot glue, visible in figure 4.10, had to be added to keep the screw in place.

Glove



Figure 4.11: Immersion's CyberGlove II

Before handing the control of the humanoid hand to a Neural Network, a natural easier step is to teleoperate it when possible. Figure 4.11 illustrates Immersion's CyberGlove II, a special glove that contains a deformable strip on each finger joint, capable of translating the muscle

flexion of the user's finger into an electric signal. Considering that the CyberGlove II can record the muscular movements the Seed manipulator can recreate, and even more, a simple Python script can encode such electric signals linearly into commanded angles for the robotic hand.

Each user has a different finger elasticity and size, while the Seed Robotics hand expects radian angle values between 0 and 2π , as mentioned above, so an initial calibration is conducted. A mapping between a full bent and a full elongated finger in the glove to their homonymous in the robotic manipulator is made to give the human expert more control over the robot. However, this does not compensate for the harsh limitations posed by Seed's hand described earlier, nor for the challenges gathering demonstrations by teleoperating the underactuated humanoid manipulator elaborated next.

4.2 Demonstration data collection

DAPG algorithm had compelling results in both simulation [3] and real [15] scenarios. The RL-aided IL method, described in detail in the previous chapter, uses demonstrations to guide the exploration of the Reinforcement Learning agent in the environment by initializing the Neural Network with a Behavior Cloning technique. In the original work, Rajeswaran et al [3] collected 25 simulated demonstrations for this task, whereas Zhu et al [15] 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. In succeeding sections, we describe briefly some in-hand manipulation trials conducted using Imitation Learning only, more precisely, Behavior Cloning trained under different amounts of demonstrations and different MLP settings.

Zhu et al [15] 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. A Python script for the Seed hand operation by Immersion's CyberGlove II, depicted in figure 4.11, was already coded as well. As one can notice, using a glove to control an underactuated manipulator is not an easy task.

Figure 4.12 shows the per-finger commanded motor values of a demonstration. The illustrated example suffered post-processing changes with respect to the number of steps, as tele-

operation for recording the demonstrations does not allow for strict control over the number of steps each trajectory has. Despite this fact, special attention was taken to have learning trajectories that did not take too much time to achieve the goal. Such action was far from effortless considering that the desired task is pretty complex and hard to complete, requiring a lot of repetitions to achieve 15 reasonable demonstrations within appropriate time frames. However, wanting to keep as many parameters as possible constant with the original DAPG work, we later scaled all our demonstrations to the same number of steps Rajeswaran et al [3] 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 4.12 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 4.12, 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° unless roughly two-thirds of its size is above the thumb's cavity bump.

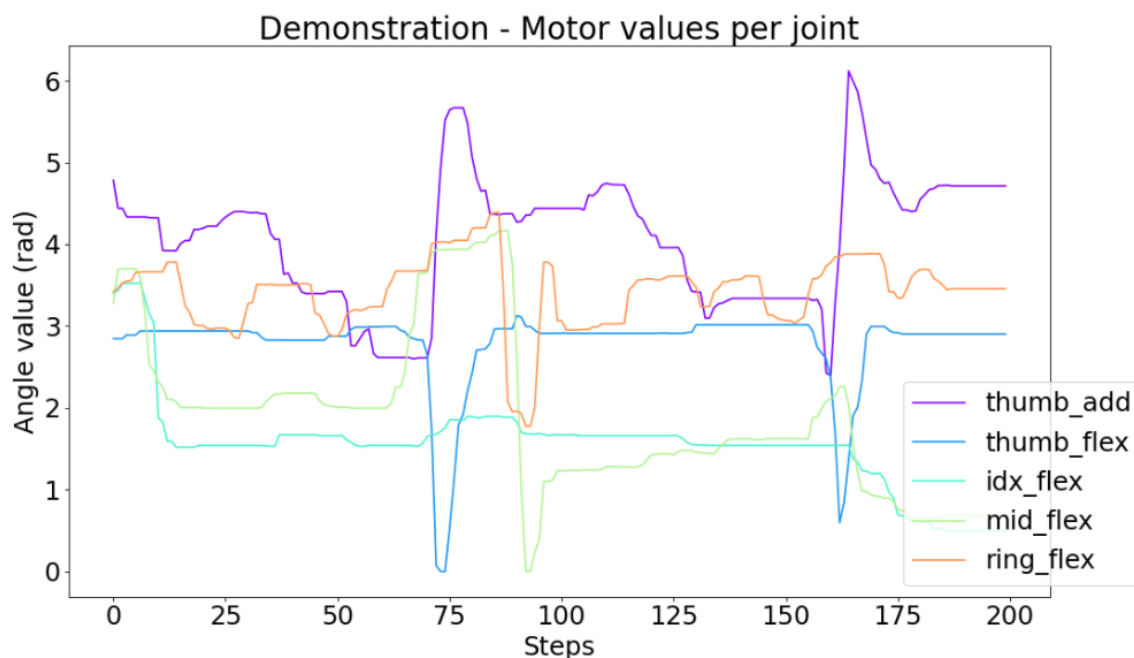


Figure 4.12: Per-finger motor commands of a demonstration

Challenges

The task our robotic hand is supposed to learn, clarified at the beginning of the chapter, is harder to accomplish than originally thought. A challenging task, rather than an unachievable one, was initially intended as a substantial test of the capabilities of our coupling of these methods to address a real in-hand manipulation situation. At first glance, reaching the 45° goal angle with the underactuated restrictions of the Seed Robotic manipulator should be demanding yet possible.

A reason why the proposed task is vastly complex has already been introduced above, with the need to use the thumb to elevate the bottle within the hand before opening the indicator and the middle fingers to allow it to tilt by gravitational forces. While collecting the demonstrations, we noticed additional critical issues. Taking into consideration all the challenges faced, we performed adjustments to the reward function as initially thought to give bonus encouragement starting at angles further away from the goal, finishing with the reward equation shown in section 3.4. Gathering demonstrations was the first factual contact we had with the full assembled framework, thus exploring all possibilities and issues of controlling the Seed Robotic and observing the bottle's pose using the Orbbec Astra camera.

- CAPTRA's pitfalls:

The problems with the pose estimation and tracking of the bottle during the in-hand manipulation assignment can be resumed in three words, lack of robustness. Preliminary tests on the pose estimation produced inspiring positive outcomes, with remarkable consistency before a never-seen object. Unfortunately, this stability on the predicted poses suffered a great deal with the occlusion created by the manipulator's fingers, as already expected and mentioned in section 3.3.1.

The theoretical advantages of employing a pose estimator and tracker, able to surpass some spontaneous occlusions, have not always worked in our favor. The embed smoothness present in object manipulation that took us to use a theoretically competent tracking estimator was not always empirically present during our learning processes. Nonetheless, CAPTRA's Neural Networks should be able to deliver enduring estimations for static pose predictions, which unfortunately was not the case. For the same initial pose parameters, corresponding to a constant initial bottle grasp, we obtained either a consistent pose of the bottle for 5 minutes or encountered along the way arbitrary pose drifts. In addition to this, sporadic random failures when fetching the RGB-D image from the Astra camera were observed, leading to pose tracking loss even when the bottle did not move at all.

While the training of the Reinforcement Learning agent was conducted in a sequential form, actuation followed by estimation, the teleoperation procedure for the gathering of

the demonstrations imposed the execution of such procedures in a parallel fashion. Naming a movement as sudden is intrinsically connected to the refreshing capabilities of the pose estimator, the so-called refresh ratio. Such movements considerably shift the bottle from the previously estimated position, most likely causing pose drifts.

The sequential format employed in the Reinforcement Learning allows limiting the difference between consecutive per-finger motor angle values to mitigate these unwanted motions. Taking into consideration that the pose is being predicted in a parallel non-real-time manner, a similar strategy would restrict the controllability of the expert severely. Already experiencing other hardware complications, the avoidance of sudden moves by the underactuated Seed Robotic hand in the teleoperated demonstrations was a slow and arduous task.

As one may expect, such a lack of robustness makes the gathering of replicable demonstrations incredibly problematic, if not impossible.

- Seed Robotic hand limitations:

An overload warning in the Seed Robotic hand indicates that the angle requested for a particular motor has not been reached after an expected internal time for it to do so. For the safeguarding of the hardware, the hand shuts down, requiring a manual reset of the manipulator. One can easily see that the existence of this protection mechanism makes total sense for situations where the fingers get stuck into something and demand too much of the motors trying to move them.

In addition to these safety warnings, the Seed Robotic hand is fortunately quite robust from a hardware point of view, as can be noticed in figure 4.7. However, as for CAPTRA's pitfalls, the same interpretation of lack of robustness can be applied to the limitations of the manipulator, where instead of random pose drifts, we receive accidental overload warnings. Accidental, in the sense that we can not control it, nor were we expecting it.

The least expected overload warning observed was finger bumping. Namely, the ring and middle fingers collide with each other when a close command to grasp the bottle arrives for both simultaneously. Commanding the bending of both fingers at the same time makes, without predictability, one of the fingers stays slightly beneath the other as a result of their collision. If the subsequent move attempts to lift the underneath finger, a situation we can not predict raised an overload error.

The underactuated hand flexes the ring finger and the pinky jointly with one motor. Nonetheless, as far as the eyes see, there is no rigid connection between these fingers

beyond the same motor bending finger's tendon wires. What happens when the motor is pulling the tendon strings but only one finger enters into contact with the 3D printed water bottle is one situation that may come to the mind of the reader. The underactuated fingers can then suffer from a misalignment crisis, where the lower fingers create an object's shape situation and, as expected, overload warnings are more likely to occur. Trying to attenuate such hurdles with the ring finger, we concentrated most of the movements on the thumb combined with the index or middle fingers.

In the end, we were able to collect 15 reasonable demonstrations containing reliable action-state pairs to train the BC initialization of DAPG's algorithm. CAPTRA's requirements for a good and consistent initial pose information, together with the challenges experienced while gathering the expert's training trajectories, took us to simplify the task to start from a repeatable starting grasp instead of random initial grasps.

4.3 Results and discussion

This section elucidates the reader to the results achieved for the whole process of this thesis, examining them against the expected outcomes. In-hand manipulation is a recent field of research, with an intensive investigation being conducted on the manner 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.

With the fast pace of growth in technology, the probed techniques to solve problems in novel fields, such as in-hand manipulation tasks, must be adaptable. Willingness to reuse works conceived in one or more of the vast areas of technology that, in the last decade, have been achieving extraordinary results is key to successful investigation in yet unexplored areas. The focus of the proposed framework was to find an uncomplicated way to provide these characteristics. One can attempt within hand manipulations tasks employing different methods from the ones selected by us.

Any big study is built upon several intermediate steps. The central task of this project, the Reinforcement Learning aided Imitation Learning teaching an Neural Network to achieve a 45° turn of a bottle using an underactuated hand, was no different. After doing an initial literature review to filter the best candidates for the job at hand, preliminary trials were performed to, step by step, approach the goal in a constructive manner. Next, we provide an analysis of such steps, from experimenting with the category-level object pose estimation on its own to the elaboration

of the iteration evaluation trials for the final RL procedure, the so-called test trials.

Category-level object pose estimation

Category-level object pose estimators attempt to recreate human’s ability to recognize and categorize instances of objects never seen before into known classes. CAPTRA was, by the time this thesis was performed, the most advanced pose predictor in this category. With huge theoretical advantages over its more immediate competitor, 6PACK, the selection was obvious. A summary of the comparison between these methods has been described above in section 3.2.1.

Prediction rate is one of CAPTRA’s drawbacks. The method falls short of real-time pose estimation, a characteristic that was believed to be crucial for smooth and quick learning of the task at hand. Apart from the challenging situations experienced with gathering the demonstrations, interleaving the object pose estimation and manipulator actuation mitigated this weakness.

An object detector instead of a core pose predictor, Google’s MediaPipe Objectron was an experimentally tried contender. MediaPipes 3D object detector is presented as lightweight and real-time for mobile applications. Moreover, Objectron offered a mug pre-trained category which could work for our task. Not being the main purpose of this thesis to quantitative document every introductory experiment, a pure qualitative evaluation of Objectron’s performance was made. The results were not satisfactory nor robust enough for the task at hand.

The first glance taken into CAPTRA’s article makes the reader believe that the provided Python code implementation already encompasses live-stream prediction mechanisms, which was not the case. We overcame this problem by creating a simple segmentation method using the capabilities of OpenCV and assembling it to CAPTRA’s core Neural Networks.

Even though we were excited by the positive results we were able to recreate using NOCS’ adapted version of CAPTRA code, we were not expecting a similar performance for our CAPTRA version on live-stream input. However, the live-stream adapted version behaved surprisingly well with the never-before-seen 3D printed water bottles. Only a pure qualitative evaluation was made at this stage once again, yet this time 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.

For reasons unknown to us, the plug-in and use installation of the Astra’s RGB-D camera in our computer was not 100% successful. The first major inconvenience is the inexplicable sporadic failure to capture and transmit the RGB-D image to the computer. The second was the puzzling nonexistence of the ROS topic related to the white balance and exposure settings. A lamp could have been placed above Astra’s camera, providing stable light conditions and

reducing possible segmentation mistakes, if the extremely high value of the camera's whites did not invalidate such a solution. The white exposure value was extremely high, invalidating this solution. The gamma decay function, described above in section 3.3.1, balanced the bottle's exposure for the color segmentation on the one hand. On another, the use of a pose tracker estimator mitigated the influence of the lapses in gathering info from the RGB-D camera.

As previously mentioned, 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 a magnificent segmentation.

Behavior Cloning

Figure 3.4 illustrates the training cycle for the in-hand manipulation task we materialized in our laboratory. The control commands actuate the Seed hand, which handles the bottle accordingly, and then the bottle's pose is estimated using the acquired RGB-D image of that moment. From the perspective of the manipulation algorithm, the input is the pose corresponding to the actuation commanded at the previous instant, and the output is the newly driven instructions for the manipulator.

In a simulation environment, such as Mujoco's physics simulator, the manipulation algorithm interacts with the simulation by either asking or giving the required pieces of information for the development of the method. The process queries directly from the simulator the required object's pose and provides to the simulator the computed actions. When the same technique is applied to the real world, the estimation and actuation become two different parts, where the second needs the first to operate.

As part of building the in-hand manipulation pipeline proposed in this thesis, conducting preparatory tests in each one of its stages would be ideal. For the visual part of the proposed system, we detailed above how such trials were performed in both the initial NOCS's version of CAPTRA and our live-stream modified version. However, a duality of these procedures for the control part of the suggested approach is not possible. The disclosed Python code implementation of the DAPG algorithm is bound to Mujoco's physics simulator. With an educational license, we were able to test with the pre-trained Neural Network in the available Mujoco scenarios to validate the desired method.

As elucidated in the preceding chapters, Rajeswaran et al [3] designed an RL-aided IL process that uses demonstrations to guide the exploration of the Reinforcement Learning agent by initializing the weights of the network with the BC technique. The most comparable manner we found to keep adding build blocks towards our goal was to split the implementation of the DAPG algorithm into two phases. The first comprised training only with Behavior Cloning, while the second introduced the RL runs.

One can argue that this idea of partitioning into Behavior Cloning plus RL lacks theoretical support. In light of the argumentation in Rajeswaran et al [3] article - "we observed experimentally that the cloned policies themselves were usually not successful." - one can even say that empirical support is absent. Even agreeing with these objections, an opportunity to corroborate these affirmations outside the simulation environment presented itself. We opted to further extend the analysis into the BC by using different quantities of demonstrations and trying small modifications to the policy network.

Going with the strategy to do these ablation studies was useful to get a palpable feeling of the actual requirements to perform the idealized in-hand manipulation task. As formerly noted, the cumbersome job of collecting 15 reasonable demonstrations threw doubts over the actual capacity of the Seed hand to thrive the demanded assignment. Despite facing these situations at such an early stage, we still took the decision to proceed with the learning of such a challenging task. After we got unsatisfactory results from the Behavior Cloning, as expected, we still pursued the desirable 45° turn goal, determined to push through the DAPG algorithm.

One could question why there is no indication of the translation of the bottle with respect to the reference frame of the manipulator in the reward function. Analogous to the bottle's angle and distance to the goal angle, and taking into account that the crucial action to successfully complete the bottle's rotation is the thumb movement that upswings the bottle above the palm's bump housing the thumbs' motor joint, the bottle's in-hand elevation could be present in the reward. In our opinion, CAPTRA's pose estimations lack the accuracy consistency required to be beneficial to integrate this data into the reward function. We believed it to be less detrimental to the learning policy to consider the noisy bottle's in-hand elevation as an unmodeled obstacle. However, we highlight this aspect as important to be studied in future research.

Given the demonstrations, teaching the BC policy network is a purely computational assignment, which allowed to experiment with nine different scenarios at a low cost. The investigation of non-obvious NN transformations is beyond the scope of this thesis leading us to go for rather straightforward modifications for these ablation studies. We did two main changes to the policy network, one in terms of its depth another in terms of its layer's size, originating three distinct MLP networks:

- The original DAPG policy Neural Network with two hidden layers, each with 32 neurons
- A one hidden layer policy Neural Network with 32 neurons
- A two hidden layer policy Neural Network, each with 16 neurons

Each one of these was then trained with 5, 10, or 15 demonstrations, originating nine distinct Behavior Cloning policies with equally insufficient performance. The first action of every BC policy was promising, as it commanded the Seed hand to turn the bottle slightly toward the goal angle. However, this was the only move each policy could achieve, as the subsequent actions computed from the new bottle state did not perceptibly change the operating motor values.

We consider the main reason behind the same poor evaluation outcomes of the Behavior Cloning policies to be the intrinsic state distribution shift problem, as substantial changes in the motor angles were observed, with the resulting bottle pose not matching any demonstration path. Other situations such as the ratio between trained epochs and trainable parameters or between the available demonstrations with the number of neurons to train could also play a substantial role in these results.

Reinforcement Learning framework and test trials

For the incorporation of the Reinforcement Learning training of the policy network using the DAPG algorithm, we went back to the original architecture. We did some toy experiments reducing the number of iterations and trajectory runs within each iteration. The observed results corroborated once again, in a more severe manner than initially thought, the affirmations of Zhu et al [15] regarding the time-expensive operation of collecting on-policy data in real-world systems. The above section 4.1 already develops a relentless analysis of the training times for this experiment, touching important the important topic of not overheating the hardware with intense straight hours of experimental work. Some workarounds and adaptations for this situation are already present.

Aware that complete training of our DAPG agent was not possible under the time restrictions for the delivery of this thesis, we came to the decision to evaluate the model at each iteration 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. During our experiments, one-tenth of the estimated by the original DAPG, the latter did not occur.

When training, random noise is added to the commanded actions in order to foment exploration. By storing the current model parameters and suppressing the noise addition, we can

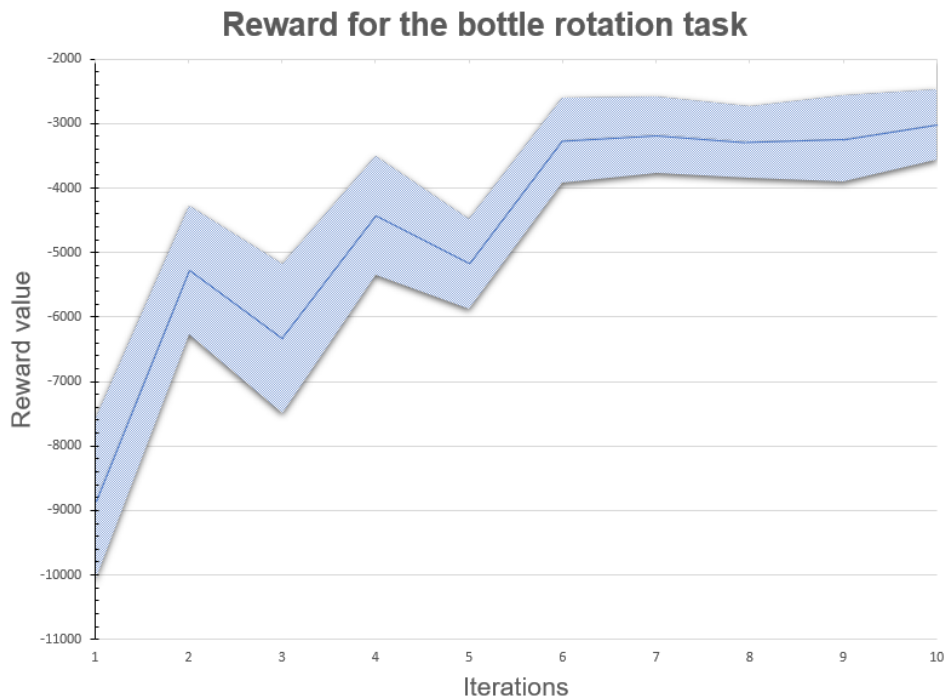


Figure 4.13: Reward progress throughout the experiments - roughly 100 hours of arm and hand motion.

alternate from training to testing the policy, keeping open the possibility to resume training where we left. 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° 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, it should deviate much.
- 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.

Table 4.1: Test trials per iteration

Test Trials		
Iteration	Mean of minimum angles (°)	Mean of final angles (°)
1	71.32	72.31
2	67.04	70.62
3	62.89	67.15
4	71.15	73.42
5 (action clipping)	73.20	75.65
6 (action clipping)	66.26	67.68
7 (action clipping)	68.8	70.40
8 (action clipping)	66.73	68.75
9 (action clipping)	66.29	70.14
10 (action clipping)	62.55	62.85
BC	76.86	76.86
BC (action clipping)	76.49	76.51

Figure 4.13 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 3.4, where the reward function used is explained, we notice that, as a general principle, the reward values are negative, aggravated if the bottle is dropped, or rewarded with bonus value if sufficiently close to the target angle. 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 4.1 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.

Skeptical in the face of the poor results attained by the Behavior Cloning, namely by the single executed movement, we developed another 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.

Table 4.2: Meaningful movements accomplished per iteration

Meaningful Movements	
Iteration	Number
1	1
2	1
3	1
4	2
5 (action clipping)	2/3
6 (action clipping)	4/5
7 (action clipping)	4/5
8 (action clipping)	5
9 (action clipping)	5
10 (action clipping)	5/6
BC	1
BC (action clipping)	2/3

Table 4.2 shows the collected results, incorporating the Behavior Cloning experimentation as well. As in any experiment, during the course of the same, situations occur that require amendments to the procedures. In our case, the need to impose a restriction on the amount of discrepancy between consecutive angle values arose halfway through the learning method, as can be seen by the indication next to the fifth iteration. While, in the RL agent, this action clipping did not affect the number of meaningful movements, for the BC agent, the situation was different, as can be seen in table 4.2. This result strongly supports that shift distribution was the main reason behind such unsatisfactory outcomes of the Behavior Cloning policy. The increase in the number of movements executed by the manipulator was not converted into better performance with respect to the final angle achieved by the bottle's rotation.

In view of the optimistic results present in figure 4.13 and table 4.1, 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 much room to maneuver, as the bottleneck is attached to the physical limitations of the actuator. Nonetheless, this thesis substantiates preceding research in RL-aided IL as potential control policies for robotic humanoid manipulators.

5 | Conclusion and future Work

5.1 Conclusions

The essays documented in this thesis look into the implementation of 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.

Reminding the query points elaborated in the section 1.2, our significant discoveries can be resumed thusly.

1. Are underactuated five-finger end-effectors sufficiently stable to perform in-hand manipulation tasks?
 - Robustness is one of the strong sides of the Seed Robotics hand used in these experiments. Such prevented several incidents from becoming more severe throughout the tests, and nowadays, this is a quality hard to find in hands with more Degree of Freedom. However, despite a positive learning curve, the hardware restrictions prevented a simple way to accomplish one of the easiest tasks for humans, a within-hand bottle rotation. We believe such underactuated manipulators are a great choice for grasping tasks, yet we must encourage others to perform complex in-hand tasks with fully dexterous manipulators when possible.
2. Are category-level object pose estimators steady enough to be incorporated within a framework, in particular an RL one?
 - CAPTRA struggled a bit with occlusions created by the moving fingers, as described in the last chapter. On some occasions, external supervision was a must to avoid completely erroneous estimations. We believe such occlusion situations are not a defect of category-level pose estimations, thus being certain we made a good choice for the visual sensor. We consider CAPTRA to be pretty solid dealing with unseen bottles, so we think category-level estimators are ready for future challenges.

3. Can the work by Zhu et al [15] be corroborated, in the sense that Demo Augmented Policy Gradient (DAPG) [3] might be a promising direction for model-free RL systems beyond simulation environments?
 - The challenging collection of demonstrations proved to be a downside of this method once again. Due to the slow data gathering in real environment experiences, a fact also highlighted by [15], we were not able to double-check DAPG against RL from scratch in our experiments. Nevertheless, we went beyond [15], with DAPG responding well to the constraints of an underactuated end-effector and, more importantly, to the provided noisy object pose estimations. Therefore, in view of our results and [82] research, we believe that Reinforcement Learning aided Imitation Learning has a bright future in this field.

5.2 Future work

Our work brings up the simplest form to couple two necessary ingredients for manipulation, perception and actuation. We then propose a few directives to carry on the work developed in this project.

- First and foremost, one can continue the experiments described in this thesis project as they are. For assistance in the reconstruction of the scenario, please see section 1.1 and appendix A.
- The perception stage of the suggested pipeline is an *off-the-shelf* method. We believe category-level 6D pose estimation to be the way to go when dealing with novel object instances [17]. Nonetheless, other approaches should likewise be tested and later compared with our results, especially top Deep Neural Network methods in hand/object pose estimation [25, 26] and occlusion datasets [19, 22]. Besides, given the panoply of pose estimation techniques available, one can easily plug in what best suits their needs.
- The actuation stage of the introduced framework is a direct implementation of previously developed algorithm [3]. We are of the opinion that Radosavovic et al [82] work deserves the opportunity to be the main player in the actuation stage, considering it to be the natural evolution of Rajeswaran et al [3] model. Plus, we consider that some refinements not present in Rajeswaran et al [3] ablation studies ought to be investigated.

Despite future real-world robotic Reinforcement Learning [53] demanding low sample complexity in their policy gradient optimizers, it could be newsworthy to try TRPO [54]

or PPO [55]. Moreover, finding ways to reset the environment on its own, as well as shaping its own reward, are the remaining ingredients that need to be explored to achieve a full and cognitive manipulation ability by humanoid robots using RL.

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 [76] 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 thesis.

Bibliography

- [1] 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.
- [2] 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)*, 2020, pp. 10 059–10 066.
- [3] A. Rajeswaran, V. Kumar, A. Gupta, G. Vezzani, 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.
- [4] A. Billard and D. Kragic, “Trends and challenges in robot manipulation,” *Science*, vol. 364, no. 6446, 2019. [Online]. Available: <https://science.sciencemag.org/content/364/6446/eaat8414>
- [5] M. T. Mason, “Toward robotic manipulation,” *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 1, no. 1, pp. 1–28, May 2018. [Online]. Available: <https://doi.org/10.1146/annurev-control-060117-104848>
- [6] D. Morrison, P. Corke, and J. Leitner, “Closing the loop for robotic grasping: A real-time, generative grasp synthesis approach,” 2018.
- [7] M. Gualtieri, A. ten Pas, and R. Platt, “Pick and place without geometric object models,” 2018.
- [8] J. Lundell, F. Verdoja, and V. Kyrki, “Robust grasp planning over uncertain shape completions,” *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Nov 2019. [Online]. Available: <http://dx.doi.org/10.1109/IROS40897.2019.8967816>

- [9] J. Mahler, M. Matl, V. Satish, M. Danielczuk, B. DeRose, S. McKinley, and K. Goldberg, “Learning ambidextrous robot grasping policies,” *Science Robotics*, vol. 4, no. 26, p. eaau4984, 2019. [Online]. Available: <https://www.science.org/doi/abs/10.1126/scirobotics.aau4984>
- [10] M. Liu, Z. Pan, K. Xu, K. Ganguly, and D. Manocha, “Generating grasp poses for a high-dof gripper using neural networks,” 2020.
- [11] X. Lou, Y. Yang, and C. Choi, “Learning to generate 6-dof grasp poses with reachability awareness,” 2020.
- [12] T. Patten, K. Park, and M. Vincze, “Dgcm-net: Dense geometrical correspondence matching network for incremental experience-based robotic grasping,” *Frontiers in Robotics and AI*, vol. 7, Sep 2020. [Online]. Available: <http://dx.doi.org/10.3389/frobt.2020.00120>
- [13] A. Depierre, E. Dellandréa, and L. Chen, “Scoring graspability based on grasp regression for better grasp prediction,” 2021.
- [14] J. R. Flanagan, M. C. Bowman, and R. S. Johansson, “Control strategies in object manipulation tasks,” *Current Opinion in Neurobiology*, vol. 16, no. 6, pp. 650–659, 2006, motor systems / Neurobiology of behaviour. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0959438806001450>
- [15] 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)*, 2019, pp. 3651–3657.
- [16] O. Kroemer, S. Niekum, and G. Konidaris, “A review of robot learning for manipulation: Challenges, representations, and algorithms,” *Journal of Machine Learning Research*, vol. 22, no. 30, pp. 1–82, 2021. [Online]. Available: <http://jmlr.org/papers/v22/19-804.html>
- [17] C. Sahin, G. Garcia-Hernando, J. Sock, and T.-K. Kim, *Instance- and Category-Level 6D Object Pose Estimation*. Cham: Springer International Publishing, 2019, pp. 243–265. [Online]. Available: https://doi.org/10.1007/978-3-030-28603-3_11
- [18] G. Du, K. Wang, S. Lian, and K. Zhao, “Vision-based robotic grasping from object localization, object pose estimation to grasp estimation for parallel grippers: a review,”

- Artificial Intelligence Review*, vol. 54, no. 3, pp. 1677–1734, Mar 2021. [Online]. Available: <https://doi.org/10.1007/s10462-020-09888-5>
- [19] S. Hinterstoisser, V. Lepetit, S. Ilic, S. Holzer, G. Bradski, K. Konolige, and N. Navab, “Model based training, detection and pose estimation of texture-less 3d objects in heavily cluttered scenes,” in *Computer Vision – ACCV 2012*, K. M. Lee, Y. Matsushita, J. M. Rehg, and Z. Hu, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 548–562.
- [20] Y. He, W. Sun, H. Huang, J. Liu, H. Fan, and J. Sun, “Pvn3d: A deep point-wise 3d keypoints voting network for 6dof pose estimation,” in *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020, pp. 11 629–11 638.
- [21] Y. Bukschat and M. Vetter, “Efficientpose: An efficient, accurate and scalable end-to-end 6d multi object pose estimation approach,” 2020.
- [22] E. Brachmann, A. Krull, F. Michel, S. Gumhold, J. Shotton, and C. Rother, “Learning 6d object pose estimation using 3d object coordinates,” in *Computer Vision – ECCV 2014*, D. Fleet, T. Pajdla, B. Schiele, and T. Tuytelaars, Eds. Cham: Springer International Publishing, 2014, pp. 536–551.
- [23] Y. Xiang, T. Schmidt, V. Narayanan, and D. Fox, “Posecnn: A convolutional neural network for 6d object pose estimation in cluttered scenes,” 2018.
- [24] Y. Li, G. Wang, X. Ji, Y. Xiang, and D. Fox, “Deepim: Deep iterative matching for 6d pose estimation,” in *European Conference on Computer Vision (ECCV)*, 2018.
- [25] F. Mueller, D. Mehta, O. Sotnychenko, S. Sridhar, D. Casas, and C. Theobalt, “Real-time hand tracking under occlusion from an egocentric rgb-d sensor,” in *2017 IEEE International Conference on Computer Vision Workshops (ICCVW)*, 2017, pp. 1284–1293.
- [26] G. Garcia-Hernando, S. Yuan, S. Baek, and T.-K. Kim, “First-person hand action benchmark with rgb-d videos and 3d hand pose annotations,” in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2018, pp. 409–419.
- [27] 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)*, 2019, pp. 2637–2646.

- [28] B. Tekin, S. N. Sinha, and P. Fua, “Real-time seamless single shot 6d object pose prediction,” in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2018, pp. 292–301.
- [29] X. Deng, A. Mousavian, Y. Xiang, F. Xia, T. Bretl, and D. Fox, “Poserbpf: A rao-blackwellized particle filter for 6d object pose estimation,” in *Proceedings of Robotics: Science and Systems*, Freiburg/Breisgau, Germany, June 2019.
- [30] 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)*, 2019, pp. 3338–3347.
- [31] X. Deng, Y. Xiang, A. Mousavian, C. Eppner, T. Bretl, and D. Fox, “Self-supervised 6d object pose estimation for robot manipulation,” in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, 2020, pp. 3665–3671.
- [32] X. Yu, Z. Zhuang, P. Koniusz, and H. Li, “6dof object pose estimation via differentiable proxy voting loss,” *ArXiv*, vol. abs/2002.03923, 2020.
- [33] Y. He, H. Huang, H. Fan, Q. Chen, and J. Sun, “Ffb6d: A full flow bidirectional fusion network for 6d pose estimation,” in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2021.
- [34] D. Lowe, “Object recognition from local scale-invariant features,” in *Proceedings of the Seventh IEEE International Conference on Computer Vision*, vol. 2, 1999, pp. 1150–1157 vol.2.
- [35] B. Calli and A. M. Dollar, “Vision-based model predictive control for within-hand precision manipulation with underactuated grippers,” in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, 2017, pp. 2839–2845.
- [36] J. Lu, F. Richter, and M. C. Yip, “Robust keypoint detection and pose estimation of robot manipulators with self-occlusions via sim-to-real transfer,” *ArXiv*, vol. abs/2010.08054, 2020.
- [37] E. Y. Puang, K. Peng Tee, and W. Jing, “Kovis: Keypoint-based visual servoing with zero-shot sim-to-real transfer for robotics manipulation,” in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2020, pp. 7527–7533.

- [38] 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)*, 2017, pp. 23–30.
- [39] S. Cruciani, B. Sundaralingam, K. Hang, V. Kumar, T. Hermans, and D. Kragic, “Benchmarking in-hand manipulation,” *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 588–595, 2020.
- [40] B. Tekin, F. Bogo, and M. Pollefeys, “H+o: Unified egocentric recognition of 3d hand-object poses and interactions,” in *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 4506–4515.
- [41] M. Oberweger, P. Wohlhart, and V. Lepetit, “Generalized feedback loop for joint hand-object pose estimation,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 42, no. 8, pp. 1898–1912, 2020.
- [42] B. Doosti, S. Naha, M. Mirbagheri, and D. J. Crandall, “Hope-net: A graph-based model for hand-object pose estimation,” in *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020, pp. 6607–6616.
- [43] P. R. Florence, L. Manuelli, and R. Tedrake, “Dense object nets: Learning dense visual object descriptors by and for robotic manipulation,” in *Proceedings of The 2nd Conference on Robot Learning*, ser. Proceedings of Machine Learning Research, A. Billard, A. Dragan, J. Peters, and J. Morimoto, Eds., vol. 87. PMLR, 29–31 Oct 2018, pp. 373–385. [Online]. Available: <http://proceedings.mlr.press/v87/florence18a.html>
- [44] L. Manuelli, W. Gao, P. R. Florence, and R. Tedrake, “kpam: Keypoint affordances for category-level robotic manipulation,” *ArXiv*, vol. abs/1903.06684, 2019.
- [45] W. Gao and R. Tedrake, “kpam-sc: Generalizable manipulation planning using keypoint affordance and shape completion,” *arXiv preprint arXiv:1909.06980*, 2019.
- [46] Y. Zhang, I. Clavera, B. Tsai, and P. Abbeel, “Asynchronous methods for model-based reinforcement learning,” in *3rd Annual Conference on Robot Learning, CoRL 2019, Osaka, Japan, October 30 - November 1, 2019, Proceedings*, ser. Proceedings of Machine Learning Research, L. P. Kaelbling, D. Kragic, and K. Sugiura, Eds., vol. 100. PMLR, 2019, pp. 1338–1347. [Online]. Available: <http://proceedings.mlr.press/v100/zhang20a.html>

- [47] A. Nagabandi, G. Kahn, R. S. Fearing, and S. Levine, “Neural network dynamics for model-based deep reinforcement learning with model-free fine-tuning,” in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, 2018, pp. 7559–7566.
- [48] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis, “Human-level control through deep reinforcement learning,” *Nature*, vol. 518, no. 7540, pp. 529–533, Feb. 2015. [Online]. Available: <https://doi.org/10.1038/nature14236>
- [49] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, “Continuous control with deep reinforcement learning.” in *ICLR (Poster)*, 2016. [Online]. Available: <http://arxiv.org/abs/1509.02971>
- [50] M. G. Bellemare, Y. Naddaf, J. Veness, and M. Bowling, “The arcade learning environment: An evaluation platform for general agents,” in *Proceedings of the 24th International Conference on Artificial Intelligence*, ser. IJCAI’15. AAAI Press, 2015, p. 4148–4152.
- [51] K. Cobbe, O. Klimov, C. Hesse, T. Kim, and J. Schulman, “Quantifying generalization in reinforcement learning,” in *Proceedings of the 36th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, K. Chaudhuri and R. Salakhutdinov, Eds., vol. 97. PMLR, 09–15 Jun 2019, pp. 1282–1289. [Online]. Available: <http://proceedings.mlr.press/v97/cobbe19a.html>
- [52] K. Cobbe, C. Hesse, J. Hilton, and J. Schulman, “Leveraging procedural generation to benchmark reinforcement learning,” in *Proceedings of the 37th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, H. D. III and A. Singh, Eds., vol. 119. PMLR, 13–18 Jul 2020, pp. 2048–2056. [Online]. Available: <http://proceedings.mlr.press/v119/cobbe20a.html>
- [53] H. Zhu, J. Yu, A. Gupta, D. Shah, K. Hartikainen, A. Singh, V. Kumar, and S. Levine, “The ingredients of real world robotic reinforcement learning,” in *International Conference on Learning Representations*, 2020. [Online]. Available: <https://openreview.net/forum?id=rJe2syrtvS>
- [54] J. Schulman, S. Levine, P. Abbeel, M. Jordan, and P. Moritz, “Trust region policy optimization,” in *Proceedings of the 32nd International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, F. Bach and D. Blei, Eds.,

- vol. 37. Lille, France: PMLR, 07–09 Jul 2015, pp. 1889–1897. [Online]. Available: <http://proceedings.mlr.press/v37/schulman15.html>
- [55] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, “Proximal policy optimization algorithms,” *arXiv preprint arXiv:1707.06347*, 2017.
- [56] J. Fu, A. Singh, D. Ghosh, L. Yang, and S. Levine, “Variational inverse control with events: A general framework for data-driven reward definition,” in *Advances in Neural Information Processing Systems*, S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, Eds., vol. 31. Curran Associates, Inc., 2018. [Online]. Available: <https://proceedings.neurips.cc/paper/2018/file/c9319967c038f9b923068dabdf60cfe3-Paper.pdf>
- [57] Y. Duan, J. Schulman, X. Chen, P. L. Bartlett, I. Sutskever, and P. Abbeel, “ rl^2 : Fast reinforcement learning via slow reinforcement learning,” *arXiv preprint arXiv:1611.02779*, 2016.
- [58] P. Abbeel and A. Y. Ng, “Apprenticeship learning via inverse reinforcement learning,” in *Proceedings of the twenty-first international conference on Machine learning*, 2004, p. 1.
- [59] C. Finn, S. Levine, and P. Abbeel, “Guided cost learning: Deep inverse optimal control via policy optimization,” in *Proceedings of The 33rd International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, M. F. Balcan and K. Q. Weinberger, Eds., vol. 48. New York, New York, USA: PMLR, 20–22 Jun 2016, pp. 49–58. [Online]. Available: <http://proceedings.mlr.press/v48/finn16.html>
- [60] J. Ho and S. Ermon, “Generative adversarial imitation learning,” in *Advances in Neural Information Processing Systems*, D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett, Eds., vol. 29. Curran Associates, Inc., 2016. [Online]. Available: <https://proceedings.neurips.cc/paper/2016/file/cc7e2b878868cbae992d1fb743995d8f-Paper.pdf>
- [61] D. Ramachandran and E. Amir, “Bayesian inverse reinforcement learning,” in *Proceedings of the 20th International Joint Conference on Artificial Intelligence*, ser. IJCAI’07. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2007, p. 2586–2591.
- [62] P. Sermanet, C. Lynch, Y. Chebotar, J. Hsu, E. Jang, S. Schaal, S. Levine, and G. Brain, “Time-contrastive networks: Self-supervised learning from video,” in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, 2018, pp. 1134–1141.

- [63] F. Torabi, G. Warnell, and P. Stone, “Behavioral cloning from observation,” in *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI-18*. International Joint Conferences on Artificial Intelligence Organization, 7 2018, pp. 4950–4957. [Online]. Available: <https://doi.org/10.24963/ijcai.2018/687>
- [64] J. Monteiro, N. Gavenski, R. Granada, F. Meneguzzi, and R. Barros, “Augmented behavioral cloning from observation,” in *2020 International Joint Conference on Neural Networks (IJCNN)*, 2020, pp. 1–8.
- [65] A. Xie, A. Singh, S. Levine, and C. Finn, “Few-shot goal inference for visuomotor learning and planning,” in *Proceedings of The 2nd Conference on Robot Learning*, ser. Proceedings of Machine Learning Research, A. Billard, A. Dragan, J. Peters, and J. Morimoto, Eds., vol. 87. PMLR, 29–31 Oct 2018, pp. 40–52. [Online]. Available: <http://proceedings.mlr.press/v87/xie18a.html>
- [66] Y. Duan, M. Andrychowicz, B. Stadie, O. Jonathan Ho, J. Schneider, I. Sutskever, P. Abbeel, and W. Zaremba, “One-shot imitation learning,” in *Advances in Neural Information Processing Systems*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds., vol. 30. Curran Associates, Inc., 2017. [Online]. Available: <https://proceedings.neurips.cc/paper/2017/file/ba3866600c3540f67c1e9575e213be0a-Paper.pdf>
- [67] J. M. Elliott and K. J. Connolly, “A CLASSIFICATION OF MANIPULATIVE HAND MOVEMENTS,” *Developmental Medicine & Child Neurology*, vol. 26, no. 3, pp. 283–296, Jun. 1984. [Online]. Available: <https://doi.org/10.1111/j.1469-8749.1984.tb04445.x>
- [68] M. Farooqu, T. Tanaka, Y. Ikezawa, T. Omata, and K. Nagata, “Sensor based control for the execution of regrasping primitives on a multifingered robot hand,” in *Proceedings 1999 IEEE International Conference on Robotics and Automation (Cat. No.99CH36288C)*, vol. 4, 1999, pp. 3217–3223 vol.4.
- [69] L. Han and J. Trinkle, “Dextrous manipulation by rolling and finger gaiting,” in *Proceedings. 1998 IEEE International Conference on Robotics and Automation (Cat. No.98CH36146)*, vol. 1, 1998, pp. 730–735 vol.1.
- [70] M. Cherif and K. Gupta, “Planning quasi-static fingertip manipulations for reconfiguring objects,” *IEEE Transactions on Robotics and Automation*, vol. 15, no. 5, pp. 837–848, 1999.

- [71] D. Rus, “In-hand dexterous manipulation of piecewise-smooth 3d objects,” *THE INTERNATIONAL JOURNAL OF ROBOTICS RESEARCH*, vol. 18, pp. 355–381, 1999.
- [72] T. Schlegl and M. Buss, “Hybrid closed-loop control of robotic hand regrasping,” in *Proceedings. 1998 IEEE International Conference on Robotics and Automation (Cat. No.98CH36146)*, vol. 4, 1998, pp. 3026–3031 vol.4.
- [73] B. Calli, A. Kimmel, K. Hang, K. Bekris, and A. Dollar, “Path planning for within-hand manipulation over learned representations of safe states,” in *Proceedings of the 2018 International Symposium on Experimental Robotics*, J. Xiao, T. Kröger, and O. Khatib, Eds. Cham: Springer International Publishing, 2020, pp. 437–447.
- [74] B. Calli, K. Srinivasan, A. Morgan, and A. M. Dollar, “Learning modes of within-hand manipulation,” in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, 2018, pp. 3145–3151.
- [75] A. S. Morgan, K. Hang, W. G. Bircher, and A. M. Dollar, “A data-driven framework for learning dexterous manipulation of unknown objects,” in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2019, pp. 8273–8280.
- [76] 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. [Online]. Available: <http://arxiv.org/abs/1808.00177>
- [77] 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.
- [78] Y. Bai and C. K. Liu, “Dexterous manipulation using both palm and fingers,” in *2014 IEEE International Conference on Robotics and Automation (ICRA)*, 2014, pp. 1560–1565.
- [79] A. Nagabandi, K. Konoglie, S. Levine, and V. Kumar, “Deep Dynamics Models for Learning Dexterous Manipulation,” in *Conference on Robot Learning (CoRL)*, 2019.
- [80] R. S. Johansson, G. Westling, A. Bäckström, and J. R. Flanagan, “Eye–hand coordination in object manipulation,” *The Journal of Neuroscience*, vol. 21, no. 17, pp. 6917–6932, Sep. 2001. [Online]. Available: <https://doi.org/10.1523/jneurosci.21-17-06917.2001>

- [81] J. Schulman, P. Moritz, S. Levine, M. Jordan, and P. Abbeel, “High-dimensional continuous control using generalized advantage estimation,” 2015. [Online]. Available: <https://arxiv.org/abs/1506.02438>
- [82] I. Radosavovic, X. Wang, L. Pinto, and J. Malik, “State-only imitation learning for dexterous manipulation,” *arXiv:2004.04650*, 2020.

A | Appendix A

A.1 Continuation of the experiments - short guide

The next steps serve as guidance for continuing the experiments described in this thesis. The code as it is already stores the trajectories every 5 runs and clears after updating the weights, so everything is ready to use in a time-friendly way. Moreover, the code is clean and with annotations that facilitate its interpretation if needed.

Physical set up

1. Add markers for the camera and the Kinova's base, with particular attention to the orientation of the base, which must be created and attached to the experimental table.
2. Hang the bottle from the ceiling in the spot reachable by the hand coupled to the arm.
3. Any structure that can provide a good color contrast with the bottle and the hand should work as a good background

Software set up

1. Using the web API, save a minimum of 2 paths for resetting and lifting the gasped bottle. Is suggested to create a path from the packed arm to the first reset position, as well from this position back to the arm's packed for easier storage. The API is friendly and doing this should be straightforward.
2. Adapt the camera intrinsics if not using an Orbbec Astra RGB-D camera
3. Change the INI-INFO matrix, in particular the translation part, to the values of the new vector that represents the distance in the 3D world between the camera's frame and the hand's frame, or Kinova's end-effector frame. This should be measured by the user.