Abstract

Robot technology is one of the pillars of modern society. Advances in information, electronic, and mechanical fields enable us to build and program machines to perform tasks in very different contexts, such as industry, surgery, and space missions.
Specifically, in manufacturing, robots are mainly used to perform repetitive and unhealthy works like assembly, welding and material handling, thanks to their mechanical robustness and ability to repeatedly perform the same movements with high accuracy and precision.

While in the early day, robot systems were constrained in isolated and known environments. Over the past few decades, robots have been asked to solve tasks in dynamic and unknown/partially-known environments, where they must **coexist** and **cooperate** with humans, while solving different **dynamic** tasks [1] (e.g. pick a requested object, whose position is not known a priori).

In this scenario, the desired characteristics of such robotic systems are: **(a) Adaptability to new conditions**, i.e., the system must be able to easily adapt to dynamic changes in system and environmental conditions, performing *"intelligent" behaviors* to handle these new scenarios and solve the desired task; **(b) Adaptability to new tasks**, i.e., the system must be able to easily adapt to both new variations of a known task and completely new tasks by exploiting experience to infer actions and solve them;

These requirements, can be challenging to achieve with traditional robot programming techniques based on hand-written policies and control methods. These conventional techniques often require a meticulous analysis of process dynamics, the construction of an analytical model, and the derivation of a control law that meets specific design criteria. This design process is tedious and time-consuming, particularly when high-level perception systems (e.g., cameras, microphones, motion sensors) are used to infer the state of the environment (such as the unknown position of a desired object relative to the end-effector) and the intentions of the human operator.

In contrast, significant advancements have been made by leveraging *learning techniques*, where the control policy is learned from data. This data can be generated either by **agent experience** [2] or by **expert demonstration** [3]. In the case of agent experience, there is a trial-and-error procedure where the control policy generates actions executed by an agent, which interacts with the environment. The parameters are then tuned according to the effectiveness of the actions, based on their impact on the environment relative to the task to be solved. In the case of expert demonstration, the control policy parameters are directly tuned using a dataset containing examples of task execution. The goal is to replicate the tasks observed in the dataset.

Specifically this thesis is framed in the context of *Learning from Demonstration* (LfD), a learning approach based on expert demonstration. According to the requirements of adaptability the thesis focus on a specific aspect of LfD named *Multi-Task LfD*. In this case, the control policy is not trained to execute a single task (e.g., picking an object) with the goal of generalizing across different objects and initial conditions [4, 5]. Instead, it is trained to handle various variations of a specific task (e.g., picking an object from different possible locations) [6] or even entirely different tasks (e.g., a single control policy that solves both picking and placing tasks as well as assembly tasks) [7, 8]. The goal is to generalize not only with respect to the objects

being manipulated and the initial conditions but also with respect to the tasks themselves. This means that by leveraging the knowledge-sharing hypothesis, we can achieve a system capable of solving new variations.

In this scenario, the learning procedure is much more challenging because we need to include and define the **conditioning signal** (i.e., the signal that informs the policy about the task to execute, the object to manipulate, and the target placing location). Additionally, the environment can contain **multiple distractor objects** (e.g., objects that can potentially be manipulated but are not of interest for a given task variation).

Regarding the conditioning signal, there are at least two intuitive approaches. The first is through a natural language description of the task to be executed [9, 10, 7], and the second is through a video demonstration [6, 8]. In the former, the task is described using phrases that specify the task details, such as "Pick the red box and place it into the first bin". Given in input the phrase, the system must be able to infer the intent of the task (i.e., the pick-place operation) and the object of interest (e.g., red-box for picking and first bin for placing), and correlate this information with the environment and robot state to effectively control the robot. In the latter, another agent (either a robot or a human operator) performs the task in a different environment configuration, records this execution, and provides the video as input to the control policy. The control policy must then infer the intent from the video (i.e., the task to be performed, the object to be manipulated, and the final state) and control the robot to complete the task according to the agent's state, the environment's state, and the commanded task. Inspired by how humans can learn to replicate tasks by simply observing their execution, the main goal of this thesis is to develop a system capable of replicating tasks shown in a video demonstration. This involves addressing challenges related to extracting task-relevant information from the video, such as identifying the

manipulated object and its final position.

Regarding the issue of distractor objects, these are generally items that are not considered in manipulation operations, which simplifies the problem significantly. However, in the context proposed in this thesis, the problem is further complicated by the fact that the semantic meaning of an object of interest or a distractor is defined at run-time by the command itself. This means that if the initial configuration consists of four objects (e.g., four boxes of different colors), a specific object may or may not be of interest based on the command given to the robot.

The main contribution of this thesis is addressing the issue of distractor objects. Specifically, it was observed that a significant problem with the methods proposed in the literature is that while the learned control policy can generate valid trajectories, allowing the robot to reach, pick, and place objects, it often manipulates the wrong object. To address this problem, two main considerations have been made:

(1) The architectures proposed in current literature predominantly consist of end-to-end architectures, which translate high-level inputs such as images into corresponding actions. Consequently, the model must acquire an implicit representation capable of encoding both the task objective and the current state of the environment, including the location of the target object; **(2)** The learning procedure optimizes a metric that is not directly linked to task success but rather aims to replicate actions similar to those of the expert, on average.

These two aspects can lead to a control policy that is not able to effectively guide the robot toward the target object. Indeed, it was observed that one of the critical points during trajectory execution is the first steps. Small errors in these initial steps can result in reaching and consequently picking the wrong object.

Based on these considerations, this thesis evaluates the possibility of developing a system that explicitly reasons about the objects of interest (e.g., target object and placing location). The control module is

then directly informed with low-level information, such as the position of the target object.

To perform this explicit reasoning, a *Conditioned Object Detector* (COD) has been developed. This module, given the video demonstration and the current agent observation as input, predicts the class-agnostic bounding box related to the target object and the final placing location. This low-level positional information is then provided to the control module, which predicts the actions to perform.

The learning procedure is then divided into two steps. The first step involves training the *Conditioned Object Detector* (COD) module, which focuses on explicitly solving cognitive tasks, such as detecting regions of interest represented by the object to be manipulated and its final location. The second step involves training the *Object Conditioned Control Policy* (OCCP), which focuses on solving the control problem using low-level positional information that can be easily mapped into the corresponding actions.

The final system has been tested in both **multi-variation single-task** scenarios and **multi-variation multi-task** scenarios. Specifically, the system was evaluated on four different tasks: Pick-Place, Nut-Assembly, Stack-Block, and Button-Press. Each task had different variations based on the manipulated object and the final state. While the tasks share common properties, they also have specific characteristics. For example, the Nut-Assembly task involves contact-rich, precise manipulation, whereas Pick-Place can be solved in a much rougher manner.

Overall, the proposed methods demonstrated very promising behaviors and a general improvement over baseline methods that do not include object-related reasoning. This shows that solving manipulation tasks with an object-oriented approach can be an effective paradigm for LfD problems. Additionally, this approach provides interpretable information to the end user, as the predicted bounding boxes can be interpreted as the locations where the robot will move.

In conclusion, this thesis addresses the problem of leveraging object priors in the context of Multi-Task Imitation Learning. The proposed methods have been tested in both simulated and real-world scenarios, demonstrating their effectiveness.

Contents

ToDo.

- ToDo -

Chapter 1

Introduction

1.1 Application context

1.2 Motivation and thesis overview

1.3 State of the art

The chapter reviews the state of the art (SoTA) on the LfD problem. Specifically, Section 1.3.1 will focus on the formalization of the LfD problem. Then, Section 1.3.3 will present various approaches and methodologies for solving the LfD problem. This will include a detailed taxonomy of the different approaches, highlighting their pros and cons, and concluding with considerations relevant to the goals of this thesis.

1.3.1 Problem formulation

In this section, the problem of Learning from Demonstration, also known as Imitation Learning (IL), will be formalized.

The core idea behind IL is to program a robot by allowing a human expert to demonstrate how to solve a specific task. This approach avoids methods that demand intricate, handcrafted rules dictating the actions of machines and the dynamics of their operating environments, which require significant time and coding expertise. To achieve this goal, a way to transfer knowledge from the expert to the robot is necessary. One possibility is to leverage *data-driven* methods that can extrapolate control rules from demonstrated trajectories [3].

In this context, we need to define two general actors: the **expert**, who demonstrates the desired behaviors, and the **learner**, whose goal is to learn and replicate the behaviors demonstrated by the expert [3, 11]. Both the expert behaviors and the learner behaviors are described in terms of **policy**, generally denoted as π^E and π^L respectively.

Since IL relies on expert demonstrations, these are collected in a dataset $\mathcal{D}^E = \left\{ \left(\boldsymbol{\tau}_i^E, c_i \right) \right\}_{i=1}^N$, where:

- $\boldsymbol{\tau}_i^E$ is the i^{th} demonstrated trajectory, generated by the expert policy $\pi^E \sim \boldsymbol{\tau}_i^E$. It can be described as:
  - A *state-action sequence*, i.e., $\boldsymbol{\tau}_i = [s_0, a_0, \ldots, s_T, a_T]$, when the ground truth action performed by the expert is available.
  - A *state-only sequence*, i.e., $\boldsymbol{\tau}_i = [s_0, \ldots, s_T]$, when the ground truth action is not available.
- c_i is the *context-vector*, containing task-related information such as the initial state of the system s_0, the position of the target object, or a representation of the task to be executed (e.g., a natural language description of the task or video demonstrations).

The state s_t can be defined in various ways. RGB images have been used in different works, either from a third-person point of view [12], representing the robot and its workspace, or from a first-person point of view with the camera mounted on the robot [13] or the gripper [10].

Additionally, 3D point-clouds generated by RGB-D images can also be used [14]. This high-level state representation can be enriched with proprioceptive signals such as joint positions, velocities, and torques [4].

Regarding the policy, the predicted action, $\hat{a}_t$, can be generated into two distinct ways. In one case, it is described as a *deterministic function*, meaning that it directly determines the action as follows: $\hat{a}_t = \pi^L(s_t, c_i)$[13]. In the other case, it takes on a probabilistic definition, where it represents a probability distribution from which to sample the desired action: $\hat{a}_t \sim \pi^L(s_t, c_i)$ [8].

According to the definitions given in [3, 15], the learner policy π^L can be defined with respect to different abstraction levels:

- *Symbolic Characterization*, the policy maps states, and context to a sequence of options, i.e., $\pi : s_t, c \rightarrow [o_1, \ldots, o_T]$, where each option is a sequence of actions. With this representation, complex tasks can be decomposed into a sequence of simple movements. However, it is hard to achieve an accurate task segmentation and motion ordering;

- *Trajectory Characterization*, the policy maps context to trajectory, i.e., $\pi : c \rightarrow \boldsymbol{\tau}$. Because it allows the initial state to be mapped to a complete sequence of actions, this representation can be used to obtain the options in the Symbolic Representation. However, they need as many dynamic features as possible, that can be difficult to obtain;

- *State-Action Characterization*, the policy maps states(-context) to actions, i.e., $\pi : s_t, c \rightarrow a_t$. This representation makes it possible to map the current state directly to the corresponding action. However, it is easy for errors to accumulate in long-term processes. The action a_t can also be defined in various ways, depending on the type of control implemented by

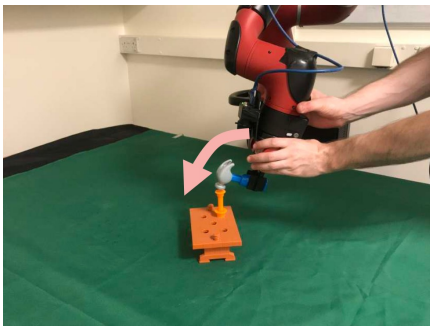
the method. Some methods operate in the joint-space, predicting motor control signals such as positions, velocities, and torques [4]. Other methods work in the operational space, where actions are assigned either an absolute pose with respect to a world frame $\mathcal{W}$, i.e., $a_i = \left[p_x^{\mathcal{W}}, p_y^{\mathcal{W}}, p_z^{\mathcal{W}}, r_x^{\mathcal{W}}, r_y^{\mathcal{W}}, r_z^{\mathcal{W}}\right]$ [8], or a displacement relative to the current gripper position, i.e., $a_i = \left[\Delta p_x^{\mathcal{W}}, \Delta p_y^{\mathcal{W}}, \Delta p_z^{\mathcal{W}}, \Delta r_x^{\mathcal{W}}, \Delta r_y^{\mathcal{W}}, \Delta r_z^{\mathcal{W}}\right]$ [13].

The expert and the learner, through their policies π^E and π^L, act on an environment modeled as a *Markov Decision Process* (MDP) [16]. An MDP is defined as a tuple (S, A, R, T, γ), where:

- $S \subseteq \mathbb{R}^n$ is the set of states (e.g., joint positions and/or images).
- $A \subseteq \mathbb{R}^n$ is the set of actions (e.g., desired end-effector pose, desired joint torques).
- $R(s, a, s')$ is the *reward function*, which expresses the immediate reward for executing action a in state s and transitioning to state s'.
- $T(s'|s, a)$ is the *transition function*, which defines the probability of reaching state s' after executing action a in state s. This distribution, which describes the *system dynamics*, can be given a priori or learned (Model-Based methods), or it may not be considered at all (Model-Free methods).
- $\gamma \in [0, 1]$ is the discount factor, expressing the agent's preference for immediate rewards over future rewards.

With respect to the given MDP definition, the reward function plays different roles depending on the approach used:

- In *Behavioral Cloning* (BC) methods, the reward function is not explicitly used. Instead, a surrogate loss function is employed.



(a) Example of kinesthetic teaching [17]

(b) Example of teleoperation [4]

Figure 1.1: Examples of direct demonstration

- In *Inverse Reinforcement Learning* (IRL), the reward function is learned, under the assumption that the expert acts (near-)optimally with respect to some unknown reward function.
- In *Generative Adversarial Imitation Learning* (GAIL) and *Learning from Observations* (LfO), the role of the reward function varies based on the specific method, as will be explained in Section TODO.

1.3.2 Source of demonstration

As stated in Section 1.3.1, IL methods rely on a dataset $\mathcal{D}^E$ of expert demonstrations. In this section, we will review the different ways to obtain these demonstrations, following the taxonomy proposed in [15].

Direct Demonstration

In the case of *Direct Demonstration*, the expert trajectory is a state-

action sequence, where the action is obtained directly from the robot. Specifically, the robot can be guided in task execution through *kinesthetic teaching* [18, 17], *teleoperation* [4, 19, 13, 7, 20, 21], or a *(hand-)written policy* [22, 6, 8, 23].

In kinesthetic teaching (Figure 1.1a), the human operator contacts and guides the robot, recording parameters such as the gripper pose, joint positions, and velocities. This has been one of the first approaches for the LfD problem [24, 25] because there is no need to consider differences in kinematics between human and robot. As a result, the data has less noise, and there is no need for expensive external tools for teleoperation. However, the robot must be passively controllable and require direct contact, which introduces safety problems and can be unintuitive for robots with multiple degrees of freedom.

In teleoperation (Figure 1.1b), the human operator remotely guides the robot with a joystick, control panel, or wearable device. These tools allow for higher safety since there is no direct contact between the robot and the human expert. Teleoperation systems have been used in various works. For example, the authors in [19, 26] proposed a teleoperation framework named Roboturk, which enables the collection of large-scale demonstration datasets [26, 5] for both simulated and real-world robots using a mobile phone as the controller. The authors in [4, 13, 7] used virtual reality controllers, allowing the human operator to intuitively move in the 3D environment, mapping the pose of the controllers to the corresponding gripper pose. This technology is of interest because it provides a safe and intuitive way to teleoperate a robot. However, the main drawback is the lack of haptic feedback, which can be mitigated by using haptic interfaces such as [27, 28].

Demonstrations collected with hand-written policies rely on the fact that the expert has access to ground truth information, such as the position of the object of interest. This assumption can be valid when a simulated environment is used to train, test, and validate the proposed methods, as seen in [6, 8, 23]. In these cases, the authors

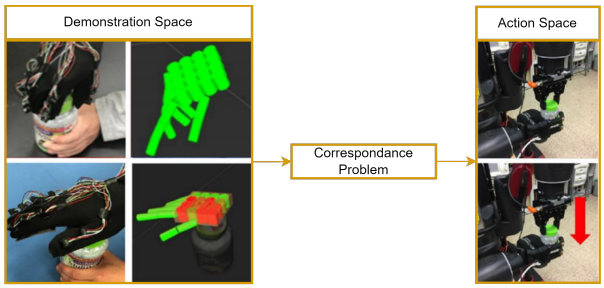
used a well-known simulation environment in the robotic learning community named Robosuite [29]. Here, demonstrations were collected to train methods using hand-written policies that have access to ground truth positional information about the object of interest, solving tasks such as Pick-Place, Nut-Assembly, Stack-Block, and more. Simulation environments facilitate the evaluation of the proposed methods and ensure reproducibility and consistent testing. The idea of using hand-engineered policies is not limited to simulation environments. Indeed, the authors in [22] used automatic grasping primitives combined with diagonal Gaussian distributions to collect demonstrations on a real-world robot. This approach aims to collect as many trajectories as possible with minimal human effort.

Indirect Demonstration

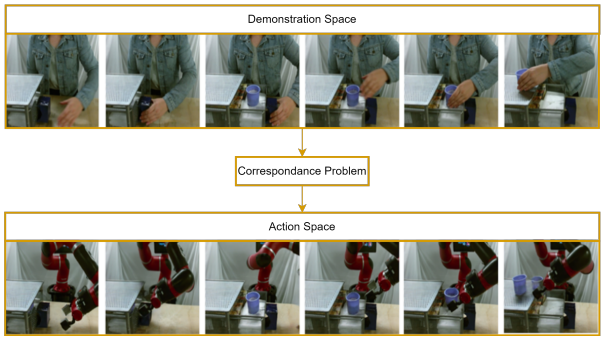
As discussed previously, in direct demonstration, an expert controls the learner agent and records the actions performed. The concept behind Indirect Demonstration (Figure 1.2) is to collect demonstrations that are completely disconnected from the target robotic platform. In the most promising scenario, a human demonstrator performs the desired tasks and records their operations. The learner, starting from this set of recordings, must be able to extrapolate the knowledge needed to replicate the observed tasks.

In this case, the expert demonstrations are state-only trajectories. Since the action space between the human demonstrator and the robot is different (consider just the different embodiment), it is not possible to directly use the human joint trajectories to minimize a supervised loss where the predicted value is related to the robot action space.

Initially, methods that follow this approach used wearable devices to capture human movement and record it [32, 30]. For example, in [32], the authors used a motion capture system to record the movement of a dancer and then transfer these trajectories to a humanoid robot.



(a) Example of indirect demonstration based on wearable device [30]



(b) Example of direct demonstration based on human video demonstration [31]

Figure 1.2: Examples of indirect demonstration

Similarly, in [30], the authors used a tactile glove [33] to record the movement performed by a human hand in the operation of opening bottles (Figure 1.2a).

In this line of research related to indirect demonstrations, there are also novel methods [31, 34, 35, 36, 37] that, inspired by the way humans learn by watching task execution, remove the assumption of having access to recorded human joint trajectories. In this case, the demonstrations are just videos of the human demonstrator (Figure

1.2b). Here, the system must infer from the video not only the intent of the task but also how this task can be solved and transform this information into its own action space.

Generally, methods based on wearable devices allow for very intuitive demonstrations, including critical information for manipulation tasks such as force and tactile information [30]. While methods based on just video demonstrations are very promising because they allow for the collection of demonstrations in the most intuitive and scalable way possible (potentially any video of a performed task can be used). However, both these approaches have to solve the *correspondence problem*, i.e., the system must be able to map motion captured in human space into the corresponding motion of the robot. In Section TODO, the different ways this problem has been solved in the context of visual demonstration will be explained in detail.

1.3.3 Learning from demonstration

This section is dedicated to presenting and analyzing various approaches for solving the LfD problem described in Section 1.3.1. Specifically, this section follows the taxonomy derived from studying different review papers [38, 39, 40, 15, 41, 11]. Figure 1.3 provides a graphical representation of the proposed taxonomy. The methods are first categorized based on the type of demonstration, either *State-Action* or *State-only*, followed by an overview of the different methodologies. The proposed taxonomy highlights the learning algorithm and main components for each methodology.

This chapter is divided into the following sections. Section 1.3.3.1 reviews methods for the fully-supervised learning methodology known as Behavioral Cloning.
Section 1.3.3.2 discusses methods under the umbrella of Inverse Reinforcement Learning, which solve the inverse optimization problem by first learning the reward function and then using it to guide policy

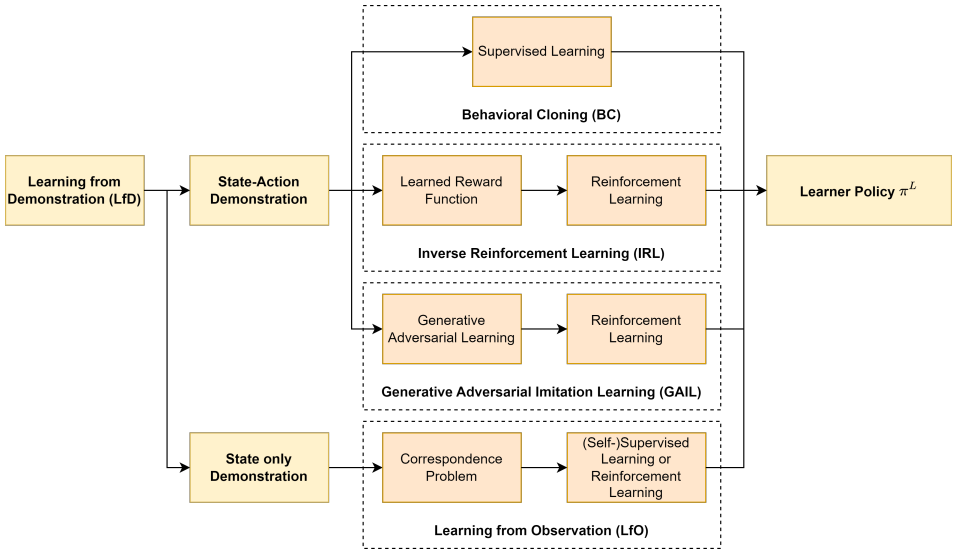


Figure 1.3: Taxonomy of LfD methods, divided based on type of demonstration and the learning algorithm used to learn the learner policy π^L

optimization following a reinforcement paradigm
Section 1.3.3.3 reviews methods leveraging the concept of *Generative Adversarial Learning* to optimize policy parameters and learn demonstrated behaviors, classified as Generative Adversarial Imitation Learning methods.
Finally, Section 1.3.3.4 covers the most recent methodology, Learning from Observation, characterized by optimizing the learner policy using state-only demonstrations.

1.3.3.1 Behavioral Cloning (BC)

Behavioral Cloning is one of the first approaches used to solve the LfD problem [42]. The high-level **supervised-learning** procedure

Algorithm 1 Abstract Algorithm for BC methods

Require: A set of expert demonstrations $\mathcal{D}^E$, a parameterized policy π_θ^L

Ensure: The optimal set of policy parameter θ^*

Optimize $\mathcal{L}$ w.r.t. policy parameter θ using $\mathcal{D}^E$

followed by BC methods is outlined in Algorithm 1. Generally, BC methods take as input the expert demonstration dataset $\mathcal{D}^E$ and a learner policy modeled as a parameterized function π_θ^L. The parameters θ can be either the weights of a neural network [42] or the parameters of a dynamic system [43]. As a supervised approach, the goal is to find the optimal parameters θ^* that can replicate the **ground-truth behaviors** contained in the dataset $\mathcal{D}^E$. This is achieved by solving an optimization problem, which can generally be described by Formula 1.1.

$$\theta^* = \underset{\theta}{argmin}\ \mathbb{E}_{(\boldsymbol{\tau},c)\sim\mathcal{D}^E}\ [\mathcal{L}((\boldsymbol{\tau},c),\ \pi_\theta^L)] \tag{1.1}$$

In the following section, different ways in which the general optimization problem described by Formula 1.1 is formulated and solved will be discussed.

Dynamical Movement Primitives

The *Dynamical-Movement Primitives* (DMPs) methods are among the first successful applications of the BC methodology to the LfD problem. Their success is attributed to their ease of implementation and efficiency in learning. DMPs do not require learning or estimating the system dynamics, nor do they require a reward function or interaction with the environment during the learning procedure, as they are supervised learning methods.

DMPs were first formalized in [43]. They derive the policy directly in the trajectory space, allowing for explicit modeling of constraints such as smooth convergence toward the goal state. The core idea behind DMPs, as proposed in [43, 44], is to model the trajectory as a **point-to-point attractor system**, described by the set of differential equations in Formula 1.2.

$$\begin{aligned} \tau\dot{y} &= \beta_s(\alpha_s(g-s)-y)+f(z) \\ \tau\dot{s} &= y \\ \tau\dot{z} &= -\alpha_z z,\ z(t)=z_0\ exp(-\tfrac{\alpha_z}{\tau}t) \end{aligned} \tag{1.2}$$

Here, β_s, α_s, and α_z are constants, s is the system state, z is the phase-variable function of time t, and f is the forcing term that describes the trajectory's non-linear behavior. Generally, f is a linear combination of basis functions $\psi_i(z)$ (e.g., Gaussian basis functions), such that $f(z(t)) = (g-s_0)\sum_{i=1}^{M}\psi_i(z(t))\omega_i z$. Essentially, a DMP describes a point-attractor system where the current system state s must converge to the goal state g, starting from s_0. In this context, the aim is to learn the set of weights $\{\omega_i, i=1,\ldots,M\}$, which can be obtained by solving a supervised learning problem with the loss function described in Formula 1.3.

$$\mathcal{L}_{DMP} = \sum_{t=0}^{T}(f_{target}(t)-f(z(t)))^2 \tag{1.3}$$

The function $f_{target}(t)$ is equal to $f_{target}(t) = \tau^2\ \ddot{s}_t^E - \beta_s(\alpha_s(g-s_t^E) - \tau\dot{s}_t^E)$ represents the evolution of the expert state s_t^E towards the goal state g, and represents the dynamic to mimic through the learned linear combination of basis function $f(z_t)$.

The initial DMPs formulation proposed in [43] has several issues that can be categorized as follows:

- **Handling stochasticity in demonstrations**: Different demonstrations can vary slightly due to differences in demonstrators, task completion methods, speeds, and paths. This variability creates a distribution in the demonstration space, requiring a method to manage it.

- **Defining different basis function**: In DMPs, the weights ω_i of the basis functions are learned. However, the force term can also be defined using other formalisms, such as Gaussian Mixture Models [45], Neural-Network Radial Basis Functions [46], or Gaussian Processes [47]. Therefore, the choice of how to model the force term is a hyper-parameter of the problem.

- **Managing arbitrary desired trajectories with intermediate via-points**: Once the behavior encoded in the demonstration is learned, generating novel trajectories that pass through new points (possibly defined by a human agent) is not possible. Therefore, a method to generalize to different waypoints is needed.

- **Handling high-dimensional inputs**: To use the DMPs algorithm, it is necessary to work in the robot space, recording joint and gripper trajectories through teleoperation or kinesthetic teaching. However, in complex scenarios involving interaction with objects that do not have fixed initial positions, it becomes essential to infer the initial object state from high-level inputs, such as images.

To address these drawbacks, several solutions have been proposed. Notably, the authors in [48] introduced the *Probabilistic Movement Primitives* (ProMPs) framework. This probabilistic framework offers an alternative movement primitive representation, capturing the variability across different demonstrations and degrees of freedom (DoFs)

through a covariance matrix. Specifically, the trajectory τ is modeled as a distribution: $\boldsymbol{\tau} = \prod_t \mathcal{N}(s(t)|\Psi(z(t))^T\omega, \Sigma_s)$, where Ψ is a time-dependent basis matrix.

Generally, modeling the problem in probabilistic terms has several advantages, particularly the ability to generalize to new goals by conditioning the learned distribution on a given novel goal state [49].

About the possibility to manage arbitrary desired trajectories, authors in [50] proposed a novel framework for learning movement primitives, named *Via-points Movement Primitives* (VMP). This is basically an extension of both DMPs (that can only adapt to new starts and goals, but cannot directly handle intermediate via-points) and ProMPs (that can adapt to via-points within the statistical distribution of the demonstrated trajectories). ToContinue

Despite all the successfully applications saw previsously, DMPs and all the variants have a very relevant limitation, that is related to the difficulty of handling high-dimensional input such as images. For this reason, the scientific community has focused the attention on methods that leverage deep architecture, that will be explained in detail in the following paragraphs.

Single-Task Imitation Learning

The *Single-Task Imitation Learning* refers to deep architecture designed to learn and replicate specific tasks from given demonstrations.

Interactive Imitation Learning

In *Interactive Imitation Learning* the learning process is augmented by interaction with a teacher, allowing for real-time feedback and adjustments to improve performance.

Multi-Task Imitation Learning

Multi-Task Imitation Learning enables the learning and execution of

multiple tasks from a set of demonstrations, highlighting the scalability and versatility of these methods.

Object-Oriented Imitation Learning
Object-Oriented Imitation Learning focuses on learning behaviors in relation to specific objects and their interactions, providing a more structured and contextual approach to imitation learning.

1.3.3.1.1 Discussion

1.3.3.2 Inverse Reinforcement Learning

This section will be dedicated to the introduction of the *Inverse Reinforcement Learning* (IRL) approach.

1.3.3.3 Generative Adversarial Imitation Learning

This section will be dedicated to the introduction of the *Generative Adversarial Imitation Learning* (GAIL) approach.

1.3.3.4 Learning from Observation

This section will be dedicated to the introduction of the *Learning from Observation* (LfO) approach.

1.3.3.4.1 Model-Free

1.3.3.4.2 Model-Based

1.3.4 Graph Neural Network for planning systems

Chapter 2

Conditioned object detector

In this chapter the COD is going to be described. Specifically, Section 2.1 will outline the detection problem being addressed. Section 2.2 will detail the proposed architecture designed to solve the described problem. Section 2.3 will discuss the experimental setup and present the results obtained from testing the proposed architecture.

2.1 Problem formulation

2.2 Architecture

2.3 Experiments

In this section, the performed experiments are going to be described. Specifically, in Section 2.3.1 the dataset used for training procedure will be described. Section 2.3.2 will report the obtained results.

2.3.1 Dataset

2.3.2 Results

This section presents the obtained results, divided into two main blocks. The first block (Section 2.3.2.1) discusses the results of the method trained to detect only the target object. The second block (Section 2.3.2.1) covers the results of the method trained to detect both the target object and the final (placing) location. For each method, results are reported for two different scenarios: first, where the method is trained in a single-task multi-variation scenario; and second, where the method is trained in a multi-task multi-variation scenario.

2.3.2.1 Target object detector

Single-task multi-variation scenario

Multi-task multi-variation scenario

2.3.2.2 Target object and final position detector

Single-task multi-variation scenario

Multi-task multi-variation scenario

Chapter 3

Object conditioned control policy

In this chapter the OCCP is going to be described. Specifically, Section 3.1 will outline the problem being addressed. Section 3.2 will detail the proposed architecture designed to solve the described problem. Section 3.3 will discuss the experimental setup and present the results obtained from testing the proposed architecture.

3.1 Problem formulation

3.2 Architecture

This section describes the entire policy architecture, specifically focusing on how the COD (Chapter 2) is integrated. There are two control architectures, differing in how the final control module predicts actions. Section 3.2.1 details the architecture that uses a single control module to predict actions for both the reaching and placing phases. In contrast, Section 3.2.2 describes the architecture that divides the con-

trol module into two distinct parts: one for computing actions during the reaching phase, and another for the placing phase.

3.2.1 Single control module

3.2.2 Double control modules

3.3 Experimental results

In this section, the performed experiments are going to be described. Specifically, in Section 3.3.1 the dataset used for training procedure will be described. Section 3.3.2 will report the obtained results.

3.3.1 Dataset

3.3.2 Results

This section presents the obtained results, divided into two main blocks. The first block (Section 3.3.2.1) discusses the results of the method described in Section3.2.1. The second block (Section 3.3.2.2) covers the results of the method described in 3.2.2. For each method, results are reported for two different scenarios: first, where the method is trained in a single-task multi-variation scenario; and second, where the method is trained in a multi-task multi-variation scenario.

3.3.2.1 Single control module

Single-task multi-variation scenario

Multi-task multi-variation scenario

3.3.2.2 Double control modules

Single-task multi-variation scenario

Multi-task multi-variation scenario

Chapter 4

Graph Neural Network for heuristic estimation

Chapter 5

Real world application

This chapter details the validation of the proposed methods in a real world scenario. Specifically, Section 5.1 describes the experimental setup. Section 5.2 discusses the dataset used to train the system. Finally, Section 5.3 presents the results obtained.

5.1 Experimental Setting

5.2 Dataset

5.3 Results

Chapter 6

Conclusions

Bibliography

[1] S. Bini, G. Percannella, A. Saggese, and M. Vento, "A multi-task network for speaker and command recognition in industrial environments," *Pattern Recognition Letters*, vol. 176, pp. 62–68, 2023.

[2] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.

[3] T. Osa, J. Pajarinen, G. Neumann, J. A. Bagnell, P. Abbeel, J. Peters *et al.*, "An algorithmic perspective on imitation learning," *Foundations and Trends® in Robotics*, vol. 7, no. 1-2, pp. 1–179, 2018.

[4] T. Zhang, Z. McCarthy, O. Jow, D. Lee, X. Chen, K. Goldberg, and P. Abbeel, "Deep imitation learning for complex manipulation tasks from virtual reality teleoperation," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2018, pp. 5628–5635.

[5] A. Mandlekar, D. Xu, J. Wong, S. Nasiriany, C. Wang, R. Kulkarni, L. Fei-Fei, S. Savarese, Y. Zhu, and R. Martín-Martín, "What matters in learning from offline human demonstrations for robot manipulation," in *Conference on Robot Learning*. PMLR, 2022, pp. 1678–1690.

[6] S. Dasari and A. Gupta, "Transformers for one-shot visual imitation," in *Conference on Robot Learning*. PMLR, 2021, pp. 2071–2084.

[7] A. Brohan, N. Brown, J. Carbajal, Y. Chebotar, J. Dabis, C. Finn, K. Gopalakrishnan, K. Hausman, A. Herzog, J. Hsu, J. Ibarz, B. Ichter, A. Irpan, T. Jackson, S. Jesmonth, N. J. Joshi, R. Julian, D. Kalashnikov, Y. Kuang, I. Leal, K. Lee, S. Levine, Y. Lu, U. Malla, D. Manjunath, I. Mordatch, O. Nachum, C. Parada, J. Peralta, E. Perez, K. Pertsch, J. Quiambao, K. Rao, M. S. Ryoo, G. Salazar, P. R. Sanketi, K. Sayed, J. Singh, S. Sontakke, A. Stone, C. Tan, H. T. Tran, V. Vanhoucke, S. Vega, Q. Vuong, F. Xia, T. Xiao, P. Xu, S. Xu, T. Yu, and B. Zitkovich, "RT-1: robotics transformer for real-world control at scale," in *Robotics: Science and Systems XIX, Daegu, Republic of Korea, July 10-14, 2023*, K. E. Bekris, K. Hauser, S. L. Herbert, and J. Yu, Eds., 2023. [Online]. Available: https://doi.org/10.15607/RSS.2023.XIX.025

[8] Z. Mandi, F. Liu, K. Lee, and P. Abbeel, "Towards more generalizable one-shot visual imitation learning," in *2022 International Conference on Robotics and Automation (ICRA)*. IEEE, 2022, pp. 2434–2444.

[9] S. Stepputtis, J. Campbell, M. Phielipp, S. Lee, C. Baral, and H. Ben Amor, "Language-conditioned imitation learning for robot manipulation tasks," *Advances in Neural Information Processing Systems*, vol. 33, pp. 13 139–13 150, 2020.

[10] O. Mees, L. Hermann, E. Rosete-Beas, and W. Burgard, "Calvin: A benchmark for language-conditioned policy learning for long-horizon robot manipulation tasks," *IEEE Robotics and Automation Letters (RA-L)*, vol. 7, no. 3, pp. 7327–7334, 2022.

[11] M. Zare, P. M. Kebria, A. Khosravi, and S. Nahavandi, "A survey of imitation learning: Algorithms, recent developments, and challenges," *IEEE Transactions on Cybernetics*, 2024.

[12] S. James, M. Bloesch, and A. J. Davison, "Task-embedded control networks for few-shot imitation learning," in *2nd Annual Conference on Robot Learning, CoRL 2018, Zürich, Switzerland, 29-31*

October 2018, Proceedings, ser. Proceedings of Machine Learning Research, vol. 87. PMLR, 2018, pp. 783–795. [Online]. Available: http://proceedings.mlr.press/v87/james18a.html

[13] E. Jang, A. Irpan, M. Khansari, D. Kappler, F. Ebert, C. Lynch, S. Levine, and C. Finn, "Bc-z: Zero-shot task generalization with robotic imitation learning," in *Conference on Robot Learning*. PMLR, 2022, pp. 991–1002.

[14] M. Shridhar, L. Manuelli, and D. Fox, "Perceiver-actor: A multi-task transformer for robotic manipulation," in *Conference on Robot Learning*. PMLR, 2023, pp. 785–799.

[15] B. Fang, S. Jia, D. Guo, M. Xu, S. Wen, and F. Sun, "Survey of imitation learning for robotic manipulation," *International Journal of Intelligent Robotics and Applications*, vol. 3, no. 4, pp. 362–369, 2019.

[16] O. Kroemer, S. Niekum, and G. Konidaris, "A review of robot learning for manipulation: Challenges, representations, and algorithms," *The Journal of Machine Learning Research*, vol. 22, no. 1, pp. 1395–1476, 2021.

[17] E. Johns, "Coarse-to-fine imitation learning: Robot manipulation from a single demonstration," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2021, pp. 4613–4619.

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

[19] A. Mandlekar, Y. Zhu, A. Garg, J. Booher, M. Spero, A. Tung, J. Gao, J. Emmons, A. Gupta, E. Orbay *et al.*, "Roboturk: A crowdsourcing platform for robotic skill learning through imitation," in *Conference on Robot Learning*. PMLR, 2018, pp. 879–893.

[20] F. Ebert, Y. Yang, K. Schmeckpeper, B. Bucher, G. Georgakis, K. Daniilidis, C. Finn, and S. Levine, "Bridge Data: Boosting Generalization of Robotic Skills with Cross-Domain Datasets," in *Proceedings of Robotics: Science and Systems*, New York City, NY, USA, June 2022.

[21] A. Mandlekar, S. Nasiriany, B. Wen, I. Akinola, Y. Narang, L. Fan, Y. Zhu, and D. Fox, "Mimicgen: A data generation system for scalable robot learning using human demonstrations," in *Conference on Robot Learning*. PMLR, 2023, pp. 1820–1864.

[22] S. Dasari, F. Ebert, S. Tian, S. Nair, B. Bucher, K. Schmeckpeper, S. Singh, S. Levine, and C. Finn, "Robonet: Large-scale multi-robot learning," in *Conference on Robot Learning*. PMLR, 2020, pp. 885–897.

[23] M. Chang and S. Gupta, "One-shot visual imitation via attributed waypoints and demonstration augmentation," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2023, pp. 5055–5062.

[24] D. Lee and C. Ott, "Incremental kinesthetic teaching of motion primitives using the motion refinement tube," *Autonomous Robots*, vol. 31, pp. 115–131, 2011.

[25] M. Saveriano, S.-i. An, and D. Lee, "Incremental kinesthetic teaching of end-effector and null-space motion primitives," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2015, pp. 3570–3575.

[26] A. Mandlekar, J. Booher, M. Spero, A. Tung, A. Gupta, Y. Zhu, A. Garg, S. Savarese, and L. Fei-Fei, "Scaling robot supervision to hundreds of hours with roboturk: Robotic manipulation dataset through human reasoning and dexterity," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2019, pp. 1048–1055.

[27] C. Systems, "Cyberforce." [Online]. Available: http://www.cyberglovesystems.com/cyberforce

[28] D. Systems, "Touch." [Online]. Available: https://it.3dsystems.com/haptics-devices/touch

[29] Y. Zhu, J. Wong, A. Mandlekar, R. Martín-Martín, A. Joshi, S. Nasiriany, and Y. Zhu, "robosuite: A modular simulation framework and benchmark for robot learning," *arXiv preprint arXiv:2009.12293*, 2020.

[30] H. Liu, C. Zhang, Y. Zhu, C. Jiang, and S.-C. Zhu, "Mirroring without overimitation: Learning functionally equivalent manipulation actions," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, no. 01, pp. 8025–8033, Jul. 2019.

[31] L. Smith, N. Dhawan, M. Zhang, P. Abbeel, and S. Levine, "AVID: Learning Multi-Stage Tasks via Pixel-Level Translation of Human Videos," July 2020.

[32] S. Nakaoka, A. Nakazawa, F. Kanehiro, K. Kaneko, M. Morisawa, H. Hirukawa, and K. Ikeuchi, "Learning from observation paradigm: Leg task models for enabling a biped humanoid robot to imitate human dances," *The International Journal of Robotics Research*, vol. 26, no. 8, pp. 829–844, 2007.

[33] H. Liu, X. Xie, M. Millar, M. Edmonds, F. Gao, Y. Zhu, V. J. Santos, B. Rothrock, and S.-C. Zhu, "A glove-based system for studying hand-object manipulation via joint pose and force sensing," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2017, pp. 6617–6624.

[34] F. Torabi, G. Warnell, and P. Stone, "Recent advances in imitation learning from observation," in *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19*. International Joint Conferences on Artificial Intelligence

Organization, 7 2019, pp. 6325–6331. [Online]. Available: https://doi.org/10.24963/ijcai.2019/882

[35] H. Xiong, Q. Li, Y.-C. Chen, H. Bharadhwaj, S. Sinha, and A. Garg, "Learning by watching: Physical imitation of manipulation skills from human videos," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2021, pp. 7827–7834.

[36] C. Wang, L. Fan, J. Sun, R. Zhang, L. Fei-Fei, D. Xu, Y. Zhu, and A. Anandkumar, "Mimicplay: Long-horizon imitation learning by watching human play," in *Conference on Robot Learning*. PMLR, 2023, pp. 201–221.

[37] Z. Qian, M. You, H. Zhou, X. Xu, H. Fu, J. Xue, and B. He, "Contrast, imitate, adapt: Learning robotic skills from raw human videos," *IEEE Transactions on Automation Science and Engineering*, 2024.

[38] L. P. Kaelbling, M. L. Littman, and A. W. Moore, "Reinforcement learning: A survey," *Journal of artificial intelligence research*, vol. 4, pp. 237–285, 1996.

[39] B. D. Argall, S. Chernova, M. Veloso, and B. Browning, "A survey of robot learning from demonstration," *Robotics and autonomous systems*, vol. 57, no. 5, pp. 469–483, 2009.

[40] A. Hussein, M. M. Gaber, E. Elyan, and C. Jayne, "Imitation learning: A survey of learning methods," *ACM Computing Surveys (CSUR)*, vol. 50, no. 2, pp. 1–35, 2017.

[41] B. Zheng, S. Verma, J. Zhou, I. Tsang, and F. Chen, "Imitation learning: Progress, taxonomies and opportunities," *arXiv preprint arXiv:2106.12177*, 2021.

[42] D. A. Pomerleau, "Alvinn: An autonomous land vehicle in a neural network," *Advances in neural information processing systems*, vol. 1, 1988.

[43] A. Ijspeert, J. Nakanishi, and S. Schaal, "Learning attractor landscapes for learning motor primitives," *Advances in neural information processing systems*, vol. 15, 2002.

[44] A. J. Ijspeert, J. Nakanishi, H. Hoffmann, P. Pastor, and S. Schaal, "Dynamical movement primitives: learning attractor models for motor behaviors," *Neural computation*, vol. 25, no. 2, pp. 328–373, 2013.

[45] W. Si, N. Wang, and C. Yang, "Composite dynamic movement primitives based on neural networks for human–robot skill transfer," *Neural Computing and Applications*, vol. 35, no. 32, pp. 23 283–23 293, 2023.

[46] J. Li, J. Wang, S. Wang, and C. Yang, "Human–robot skill transmission for mobile robot via learning by demonstration," *Neural Computing and Applications*, vol. 35, no. 32, pp. 23 441–23 451, 2023.

[47] Y. Fanger, J. Umlauft, and S. Hirche, "Gaussian processes for dynamic movement primitives with application in knowledge-based cooperation," in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2016, pp. 3913–3919.

[48] A. Paraschos, C. Daniel, J. R. Peters, and G. Neumann, "Probabilistic movement primitives," *Advances in neural information processing systems*, vol. 26, 2013.

[49] M. Saveriano, F. J. Abu-Dakka, A. Kramberger, and L. Peternel, "Dynamic movement primitives in robotics: A tutorial survey," *The International Journal of Robotics Research*, vol. 42, no. 13, pp. 1133–1184, 2023.

[50] Y. Zhou, J. Gao, and T. Asfour, "Learning via-point movement primitives with inter-and extrapolation capabilities," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2019, pp. 4301–4308.

List of Figures

List of Tables