IMPERIAL

On Self-Supervision and Foundation Models in Robot Learning

Norman Di Palo

2025

Department of Computing

Imperial College London

Submitted in part fulfilment of the requirements for the degree of

Doctor of Philosophy at Imperial College London

Abstract

Robotics unlocked an era of unprecedented automation in the creation and manufacturing of goods and services. However, the vast majority of robots are currently confined into industrial plants, with a hand-defined behaviour they repeat, and unable to deal with the uncertainties and difficulties of the real world. Equipping robots with the same dexterity, common-sense, and skilful interactions that humans exhibit in their environments, such as homes and offices, could unlock orders of magnitude more tasks where robots could cooperate with humans and substantially reduce the cost and scarcity of many goods and services.

Traditional robotics techniques are insufficient to develop the vast repertoire of skills required for human-like interaction. The recent literature has therefore adopted tools and techniques from artificial intelligence and machine learning, where algorithms are no longer manually designed to perform tasks, but instead a general learning algorithm is employed that can learn to emulate the desired behaviour and demonstrate emergent abilities. This led to the popularisation of the field of robot learning. However, the main current bottleneck is the need for large datasets that these methods generally require to obtain mastery of the desired behaviours.

In this thesis, we focus on a set of techniques and paradigms that can tackle the data hunger of large machine learning models. In particular, we focus on two main families of methods: 1) self-supervised learning and data collection methods, that allow robots to collect data autonomously, strongly reducing the amount of human time needed to learn new tasks, and 2) methods based on existing foundation models, that can transfer the abilities of these models, trained on non-robotics specific data, to robotics applications.

Originality and Copyright Statement

This thesis is the product of my own work and is the result of my own research, except where it is otherwise indicated. Some chapters of this thesis appeared as stand-alone papers in publications and conferences over the past years: this is clearly outlined in Chapter 1. The copyright of this thesis rests with the author. Unless otherwise indicated, its contents are licensed under a Creative Commons Attribution-Non Commercial 4.0 International Licence (CC BY-NC). Under this licence, you may copy and redistribute the material in any medium or format. You may also create and distribute modified versions of the work. This is on the condition that: you credit the author and do not use it, or any derivative works, for a commercial purpose. When reusing or sharing this work, ensure you make the licence terms clear to others by naming the licence and linking to the licence text. Where a work has been adapted, you should indicate that the work has been changed and describe those changes. Please seek permission from the copyright holder for uses of this work that are not included in this licence or permitted under UK Copyright Law.

Contents

1 Introduction and Thesis Contributions **19**

2 Society and Robots: The Ideas and Events that Shaped the Field **25**

2.1 Part 1: What to Optimise and Meta Problems 25

2.1.1 The Big Picture 25

2.1.2 Meta Solutions and the Rise of Automation 27

2.1.3 Too Many Meta Problems: the Long Tail of the Mundane 28

2.1.4 Abstractions All The Way Down: Intelligence as the Meta^2 Solution . . . 29

2.2 Part 2: A Brief (and Shared) History of Robotics and Artificial Intelligence . . . 31

2.2.1 Experts and Search: the First AI Wave 31

2.2.2 Intuition through Experience: the Rise of Reinforcement Learning 34

2.2.3 The Unreasonable Success of Scaling: Intelligence from Data 37

3 A Gentle Introduction to the Ideas of Robot Learning: Theoretical and Literature Background **41**

3.1 Acting: A Brief Introduction to the Kinematics of Robot Arms 42

3.2 Seeing: Modelling Cameras as Robot Eyes 44

3.2.1 Beyond RGB: Measuring Depth with modern cameras 46

3.2.2 Visual Servoing: Connecting Vision to Action 46

3.3 Communicating: Representing Language as Bits 47

3.4 Learning: Finding Sense from Data 48

3.4.1 Neural Networks: A Brief History, and What Made Them Work 49

3.4.2 Architectures Used in this Work 53

3.5 Imitating: Learning Behaviour from Data 57

3.5.1 Early Successes 57

3.5.2 From Pixels to Actions: End-to-End Neural Computing 58

3.5.3 Multi-Task, Multi-Modal Robots 59

3.5.4 Transferring Knowledge Across Tasks: Building on Foundation Models . 60

4 Learning Multi-Stage Tasks with One Demonstration via Self-Replay 62

4.1 Introduction . . . 62
4.2 Related Work . . . 64
4.3 Method . . . 65
4.4 Experiments . . . 70
4.5 Limitations . . . 79
4.6 Discussion . . . 79

5 On the Effectiveness of Retrieval, Alignment, and Replay in Manipulation 85

5.1 Introduction . . . 85
5.2 Related Work . . . 87
5.3 Method . . . 88
5.4 Experiments . . . 94
5.5 Limitations . . . 100
5.6 Discussion . . . 100

6 DINOBot: Robot Manipulation via Retrieval and Alignment with Vision Foundation Models 102

6.1 Introduction . . . 102
6.2 Related Work . . . 104
6.3 Method . . . 105
6.4 Experiments . . . 108
6.5 Limitations . . . 115
6.6 Discussion . . . 115

7 Keypoint Action Tokens Enable In-Context Imitation Learning in Robotics 117

7.1 Introduction . . . 117
7.2 Related Work . . . 119
7.3 Method . . . 121
7.4 Experiments . . . 127
7.5 Limitations . . . 139
7.6 Discussion . . . 140

8 R+X: Retrieval and Execution from Everyday Human Videos **141**

8.1 Introduction 141

8.2 Related Work 143

8.3 Method 145

8.4 Experiments 154

8.5 Limitations 162

8.6 Discussion 162

9 Conclusion **163**

List of Figures

1.1 The behaviour of machines around us has evolved over the last decades, and the rapid improvement of learning algorithms has modelled it to appear closer to human behaviour. At the same time, these machines face increasingly unpredictable environments, and can respond with equally unpredictable behaviour. Images taken from computerhistory.org, covariant.ai/robotic-induction/, blog.google, xiaomi.com, zdnet.com, ami.iit.it/human-robot-collaboration. 21

2.1 Two pivotal moments for AI involved a boardgame match against a top ranked professional player. The victories of the AI systems were unprecedented, but for two different reasons, as we describe in this chapter. Figures taken from independent.co.uk, newyorker.com. 34

2.2 These figures show the increase in computing power, measured as FLOPs, used on notable models over the decades. The right figure focuses on the models released after 2015. Figures taken from https://epoch.ai/blog/compute-trends. . 37

3.1 A minimalist illustrations of the pinhole camera model, highlighting the role of the extrinsic parameters, that define the pose of the camera in the world, and the intrinsic parameters, that define how an object will appear on the image plane. 45

3.2 A minimalist illustrations of the core layers that define the three main network architectures used in this work. The illustration serves to elucidate the structural relations between the input dimensionality and the output dimensionality of these layers. Each layer is composed of the inputs on the left, the mapping operation in the center, and the outputs on the right. 54

3.3 Top: Examples of users teleoperating robots. On the left (from [219]), a user teleoperates a robot through a Virtual Reality (VR) headset and controllers. On the right (from [210]), a user teleoperates a robot using kinematically similar arms, effectively puppeteering the robot that will mirror either the desired gripper poses, or each joint angle. Bottom (from [210]): a series of common controllers to teleoperate robots. 58

3.4 Robots that operate in human spaces will encounter complex environments, and will need to understand nuanced instructions. Image taken from deepmind.google. 60

4.1 An example of a two stage task: picking up a knife, and then using it to cut a toy vegetable. 62

4.2 An example of the training steps for a two-stage task: picking a cup and placing it on a plate. The human operator brings the robot to the *bottleneck pose* on top of the cup (1), and records the object interaction phase by grasping the cup and lifting it (2). The operator then brings the robot on top of the plate (3), and records the second object interaction phase (4), by bringing down the cup and releasing it. The operator then resets the environment by placing back the cup. The Self-Replay phase starts: the robot gathers data to learn how to reach $s_{b,0}$ exploring the workspace (5). When the data collection phase ends, the robot moves to $s_{b,0}$ and replicates the first object interaction by executing the recorded actions A_0 (6). It then repeats these steps for the second stage (7,8). We suggest to watch the video on our website for a full example. 65

4.3 A visual description of the method solving novel configurations of a task at test time. Each stage is solved by first reaching the bottleneck pose, and then replaying the interaction actions. The image is passed through all the networks described in 4.3. The videos on our website provide additional examples of this. 67

4.4 The set of tasks we tested our method on. 71

4.5 General neural network architecture used in this work, inspired by [191, 65]. We show in this figure the policy network π. The other networks have an almost identical structure, with the only difference being the last layer of the fully-connected network. 81

4.6 In these plots, we compare the time efficiency against task solving performance of Self-Replay and various baselines. 81

4.7 In these plots, we compare the sample efficiency against task solving performance of Self-Replay and various baselines. 82

4.8 An example of the robustness of our method to visual distractors. From left to right: original image, network reconstruction, and visualisation of the keypoints autonomously learnt by the network. The robot is trained to pick up the hammer and hit the green target, and the keypoints-based Deep Spatial Autoencoder learns to focus on those objects. By adding distractors (one more for each row), the reconstruction stay mostly unchanged, showing that the network is not confused by the unseen objects. In the last row, me move the task's objects, and can see how the reconstruction is able to follow that movement, hence showcasing how the network is able to predict the objects' poses. It can also be seen that the keypoints remain mostly unchanged: this shows that the network output is not influenced by distractors, and becomes a function only of the task's objects. We strongly suggest to watch the video on our website that better visualizes how the keypoints learn to track only the task's objects. Additional videos on our website demonstrate the method robustness to these visual distractors. 83

4.9 An example of the Hammer task with additional distractor objects. Additional videos on our website demonstrate the method robustness to these visual distractors. 84

5.1 An overview of the framework we study, showing the *retrieval*, *alignment*, and *replay* phases. Together, these enable one-shot imitation learning without prior object knowledge, as well as generalisation to novel objects and novel classes. . . 86

5.2 We collect a single demonstration per object following the steps in Sec. 5.3. Through autonomously collected data, our method trains a goal-conditioned alignment policy, that will align the robot to the bottleneck pose before executing the trajectory. All collected data is stored in the memory buffer. 89

5.3 Overall illustration of our framework. Upon observing a new object, the agent visually compares it to other objects it observed during training. After finding the most similar object, the agent retrieves a goal observation of that object to guide the visual alignment phase (*where* to interact), and also a trajectory to execute once the alignment is complete (*how* to interact). 92

5.4 A taxonomy of the different robot manipulation paradigms we study and examples from recent literature, based on the concepts of retrieval and/or decomposition. 96

5.5 Performance of our method and baselines on a series of manipulation tasks involving tens of different objects, with the average performances as a dashed line. 97

5.6 Average success rate of our method and the baselines, where each is provided with either (1) or (10) demos per training object. 99

6.1 From a single demo, DINOBot can learn to adapt to new objects, be robust to distractors, execute multi-stage long horizon tasks, and interact with complex environments. 103

6.2 Overall illustration of our framework. Upon observing a new object, the robot visually compares it other objects observed during demonstrations to find the most similar object (*semantic, image-level reasoning*), and retrieve both its image and the trajectory executed on that object. Then, the robot aligns its end-effector with this image (*spatial, pixel-level reasoning*), before then executing that trajectory. These two phases of reasoning are both based on extracting and matching DINO-ViT features. 104

6.3 In each column, given a live image (top) DINOBot retrieves from the buffer the most similar image in the buffer (bottom), and finds correspondences between the two. 106

6.4 Success rates on each object for all methods, tested with 10 episodes per object. Here we also report the average success rate with 95% confidence Wilson intervals:
Training set objects: DINOBot: 0.74 [0.68, 0.80], **DOME**: 0.61 [0.54, 0.67], **BC-DINO**: 0.14 [0.10, 0.19], **VINN**: 0.14 [0.10, 0.20].
Test set objects: **DINOBot**: 0.71 [0.66, 0.76], **DOME**: 0.43 [0.37, 0.49], **BC-DINO**: 0.11 [0.08, 0.15], **VINN**: 0.16 [0.12, 0.21] 108

6.5 Success rates of our method and several baselines on complex kitchen tasks. Here we also report the average rate with 95% confidence Wilson intervals.
DINOBot: 0.65 [0.50, 0.78], **BC-DINO**: 0.17 [0.09, 0.32], **VINN**: 0.40 [0.26, 0.55] 111

6.6 Despite the distractors, sensible correspondences can still be found. 112

6.7 Comparison of retrieval accuracy of the baselines, with 95% confidence Wilson intervals. 113

6.8 Alignment accuracy benchmark: we measure average translation (in pixels) and rotation (in degrees) error at finding the correct translation and rotation between a goal and a current observation. We also show 1 standard deviation interval. . . 113

6.9 Success rates for the one-shot, intra-class IL generalisation experiments, where we provide a single demo on an object and test generalisation both to the same and to unseen objects of the same class.

We also provide average success rates aggregated over all 4 objects classes, with 95% confidence Wilson intervals.

DINOBot 0.78 [0.72, 0.83], **DOME** 0.36 [0.30, 0.43], **BC-DINO** 0.06 [0.03, 0.10], **VINN** 0.10 [0.06, 0.14] 114

6.10 Performance of each method on tasks that require adaptability, dexterity or precision. We show average success rate and 95% confidence Wilson intervals. . 115

7.1 We introduce Keypoint Action Tokens (KAT), a method to repurpose off-the-shelf, large text-pretrained Transformers for efficient imitation learning, by representing sequences of observations and actions as sequences of keypoint-based tokens, converted into text and added to a prompt. KAT can learn general behaviours with as few as 10 demonstrations, and since no training is needed, learnt skills can be deployed immediately after the demonstrations. 117

7.2 An illustration of our pipeline. KAT transforms a visual observation into a *sequence of keypoint tokens*. From it, the text-pretrained Transformer (LLM) predicts *action tokens*, that are then executed as a trajectory of poses. 118

7.3 Illustration of the extraction pipeline of *keypoint tokens*. 120

7.4 Illustration of the *action tokens*, used in this work to represent $SE(3)$ end-effector poses. 124

7.5 The tasks we evaluated our method and the baselines on. 127

7.6 Success rate of each method as a function of the number of demos. While KAT outperforms the baselines in the few-shot regime (≤ 20 demos), in-context learning struggles to improve as the number of demos increase even more. Plot shows mean and standard deviation across tasks. 132

7.7 Success rate of our method as a function of the number of keypoints extracted. Plot shows mean and standard deviation across tasks. 133

7.8 Success rate of our method when changing the vision backbone model from which we extract *keypoint tokens*. Plot shows mean and standard deviation across tasks. 134

7.9 Success rate of our method as a function of the number of action tokens extracted from each demonstration trajectory, that strongly correlates with the number of action tokens generated at test time. Plot shows mean and standard deviation across tasks. 135

7.10 Success rate of our method with different representations of the end-effector poses. Plot shows mean and standard deviation across tasks. 135

7.11 Detailed comparison of KAT and Diffusion Policies interpolation and extrapolation abilities on a 2D *draw circle* task. On the left, we show where the 9 demos were given, and binned performance on 100 test cases. On the right, we show a few example test cases, demonstrating also visually the difference in the trajectories generated by KAT and Diffusion Policies. In red we show the keypoints extracted from the randomly sampled circle. The trajectories are colored from yellow to green. 137

7.12 Success rate of KAT by varying the underlying LLM acting as the imitation learning machine. Plot shows mean and standard deviation across tasks. 138

8.1 Given a language prompt, R+X first retrieves short relevant video clips extracted from a long unlabelled video of a human performing everyday tasks, recorded with a wearable camera. With the use of the retrieved video clips and a VLM, R+X performs in-context imitation learning allowing it to immediately generate and execute the desired behaviour on the robot. Videos are available on our website https://www.robot-learning.uk/r-plus-x. 141

8.2 The main assumptions and constraints of many recent learning from observation methods, Vid2Robot [106], WHIRL [17], DITTO [91], ScrewMimic [16], Orion [222], DexCap [204] 143

8.3 Upon receiving a language command, R+X retrieves all the relevant clips from the human video. Each retrieved clip is transformed from pixels to a sparse 3D points representations of the hand joints movement and salient parts of the visual observation. 145

8.4 The visual 3D keypoints of the first frame of each of the Z videos obtained from the retrieval phase along with each extracted hand joint trajectory are used as context for KAT. To execute a skill, visual 3D keypoints are extracted from the live observation and used as input to KAT which generates a sequence of hand joints. By mapping this sequence to gripper poses the robot executes the desired task. 146

8.5 Visual keypoints are extracted autonomously from RGB via a matching algorithm [9]. 147

8.6 We test R+X on 12 everyday tasks, executed by a human user in different rooms and with different distractors. 149

8.7 A short segment of our recorded long, unlabelled video of a human performing everyday tasks. The frames marked with red correspond to frames where no human hands were detected, while the frames marked with green correspond to frames where HaMeR [156] detected human hands. In practice, we discard the frames where no human hands are present and only retain a long video of frames concatenated together depicting only the human interacting with various objects. 150

8.8 An example of keypoints being tracked in a retrieved video clip. Notice how, thanks to the segmentation model [135], we can sample keypoints that stick to static parts of the scene, like the table. 151

8.9 Examples of different human hand interactions and extracted gripper actions based on three different heuristics for grasping, pressing, and pushing. The blue dots in the third column correspond to the 21 hand joints extracted using HaMeR [156]. Which heuristic to use is determined by Gemini at the Retrieval phase based on Gemini's understanding of the language command provided by the human. Note that for the pressing and pushing task the gripper is rendered as open, but in practice the gripper is closed as if it grasping something. 153

8.10 Examples of spatial, language and distractors generalisation experiments. Gripper trajectories move from red to blue. Plots provide success rates of each method on each generalisation experiment. Here we also report average success rates and 95% confidence Wilson intervals.
Hard Language Generalisation: R+X 0.60 [0.42, 0.75], **R3M** 0.03 [0.01, 0.17], **Octo** 0.00 [0.00, 0.11]
Hard Spatial Generalisation: **R+X** 0.75 [0.53, 0.89], **R3M** 0.45 [0.26, 0.66], **Octo** 0.35 [0.18, 0.57],
Distractors Generalisation R+X 0.75 [0.53, 0.89], **R3M** 0.70 [0.48, 0.85], **Octo** 0.65 [0.43, 0.82] 158
8.11 Left (barplots 1-3): We compare R+X and the baselines' ability to learn tasks in succession, and the time needed to learn such new tasks. Rightmost barplot: Gemini's retrieval performance. 160
8.12 Examples of keypoints extracted for the same tasks, but with different views, settings, and target objects. Images are grouped in pairs column-wise. Keypoints are extracted between top and bottom images of each column. 161
8.13 Success rate as a function of the amount of retrieved videos. We plot mean and standard deviation on 4 tasks, described below. 161

List of Tables

4.1 Percentage of successes of the variants of our method, Random Self-Replay (R. S-R) and Active Self-Replay (A. S-R) and a series of baselines on 20 configurations of each tasks. Experiments are repeated over 3 random seeds and show mean and 1-std. 73

4.2 Rate of successes and 95% confidence Wilson intervals of our method against an end-to-end behavioural cloning method, Relay Imitation Learning, with 10 or 30 demonstrations, over 20 configurations of each task. We measure how performance changes as the number of stages of a task increases. 75

4.3 Rate of successes in the presence of visual distractors in the environment. We also include average success rate with 95% confidence Wilson intervals. These results show the presence of distractor does not strongly impact the performance of our method. 78

4.4 Rate of successes of replicating the operator's actions during the interaction phase against using a policy learnt from demonstrations. We also include average success rates with 95% confidence Wilson intervals. These results show that our replay method, using 1 demo, achieves the same performance as a policy trained with 10, when starting from the pose reached after the alignment phase. 79

6.1 Success rates with and without distractors, with 95 % confidence Wilson intervals. 111

7.1 Success rates with 10 demonstrations for each task, for KAT and the baselines (best in bold). We also provide 95% confidence Wilson scores on the aggregated average success rates. 128

7.2 Success rates for our method showing the effect of distractors and different background (BG). 133

7.3 Numerical error on *draw circle* task of Diffusion Policy and KAT, highlighting interpolation, extrapolation and general performance. Lower values are better. . 138

8.1 Result of the various methods on the 12 proposed tasks. We report success rates over 10 episodes per task, and also 95% confidence Wilson intervals over the aggregated average success rates. 157

How to Read this Thesis

This work contains a list of papers that resulted from years of research. While they are all connected by a common research goal, each chapter is self-contained and can be understood in isolation from the others, given enough background knowledge of the field. Therefore, a reader that is only interested in a certain topic can read only a subset of the chapters.

The introduction chapters (1 and 2), while also optional, provide an overview of what is the role of robotics in society, how it is undergoing a fundamental paradigm shift, and how the works presented in this thesis are steps towards this direction.

A complete and rigorous mathematical and technical introduction to robotics, machine learning, and all related subjects is beyond the scope of this work. We will however introduce in Chapter 3 the set of ideas from mathematics, machine learning, computer science and optimisation that are related to and adopted in our research efforts, in order to help the reader both understand the landscape of robotics and artificial intelligence (AI) research and see how this thesis' contributions are relevant to this vast scenario. The author therefore assumes the reader has a basic understanding of the above topics, and will introduce the concepts in a way that will help intuitive understanding and the creation of mental links among all the concepts.

Chapter 1 is a short introduction to the field of robot learning, ideal for the reader that is familiar with the field, and lists the technical contributions of this thesis. Chapter 2 goes more in depth into *why and how* we do robot learning, and can be a useful read for the interested reader who wants a longer overview of the field, elucidating what events shaped it and brought us to the current state of research. It describes *why* our research effort is worth exploring under the lens of benefiting humanity. It also provides a snapshot of the last decades of research, and the main breakthroughs that shaped the field as it is now, describing *where* we are now. Chapter 3 offers a gentle introduction to the mathematical and technical tools needed to understand the following chapters, also focusing on the literature background on top of which we built our contributions, somehow defining *what* we talk about, establishing the basics of a mathematical vocabulary for the chapters to come. Each subsequent chapter is then focused on a different

paper, results of the research efforts during these past years, and its technical contributions to the field. A conclusion chapter then ties up the thesis and provides a broader view of the results we achieved and of future work.

We can start.

Chapter 1

Introduction and Thesis Contributions

This thesis focuses on the field of robot learning. Robot learning is a relatively recent discipline at the intersection of machine learning and robotics. More detailed definitions will be given in the following chapters; however, the fundamental minimum to comprehend the goal of this line of research is the following: we study the problem of *making robots learn tasks that would be too difficult or time-consuming to manually design a solution for.*

This concept is possibly the most important paradigm shift between classical robotics and robot learning, and that has more broadly sparked a fundamental change in many areas of computer science. It is undeniable that, in recent years, machine learning has obtained results that have dramatically raised the bar of what computers can do, and most importantly what computers can *learn*. Machine learning is not a very recent field per se, but it has often been met with scepticism by researchers and scientists over the last decades, doubting that simple learning algorithms could scale beyond trivial problems. The next chapter will provide a more detailed history of the field, but it is sufficient to mention now that all breakthroughs came from the intersection of large machine learning models, enormous datasets, and unprecedented computational power. When these conditions were met, machine learning algorithms exceeded all expectations. In particular, machine learning was seen as the optimal solution for a particular family of problems: the ones for which designing a solution is impractically difficult, but for which providing examples of solutions is considerably easier, and possibly more scalable[1].

We can now more clearly see the connection with the motivation for robot learning we mentioned as the beginning of the chapter: making robots learn tasks that would be too difficult or time-consuming to manually design a solution for. Historically, robots operated in

[1]The concept of scalability is a fundamental one. Coordinating and efficiently parallelising the work of thousands of engineers designing a solution for a task is operationally challenging. Coordinating a thousand workers giving examples of solutions for the same task is substantially easier.

very engineered environments, such as factories, where their interaction with objects and their surroundings was carefully hand-designed by expert programmers. However, this approach cannot scale beyond the metaphorical and physical walls of factory floors: designing robot policies that could work in an arbitrary human environment is intractable. A good comparison that elucidates this problem is self-driving cars. It is possible to design algorithms that can drive a car around a specified race circuit. By taking into account the shape of the track, visual cues that are known to be present, and possibly an array of sensors that can provide the position of the car in space, the task can be tackled within a reasonable time frame. However, the same approach cannot be applied to a car designed to drive on any given urban road. The unpredictability of human environments, even when limited to driving, is too complex to model analytically. And yet, we are starting to see cars able to drive themselves: the solution, unsurprisingly, is machine learning. The engineering challenge has shifted from analysing the environment and designing an artificial behaviour, to designing a system that can collect data from human drivers, and learn all the minute details from it, that can eventually lead to an artificial driver you would feel comfortable giving the wheel to (Figure 1.1 provides an overview of the evolution in behaviour of machines[2]).

This is the same paradigm shift that differentiates classic robotics and robot learning, as we mentioned. The research community recognised the inherent intractability in manually designing solutions for the vast spectrum of everyday tasks robots must perform (an unseen intersection for a self-driving car in West London is what grasping *that* pink mug from the dishwasher is for a robot). Instead of *programming* how to perform tasks, *demonstrating* robots how to execute such tasks is vastly easier. This shifts the core of the research efforts from the explicit design of behaviours to the design and implementation of machine learning pipelines[3]. The goal of the field of robot learning is to replicate the success stories of machine learning seen in other fields.

What prevented robot learning from achieving the same unprecedented results is the lack

[2]The evolution of technology has often correlated with the increase in simplicity in the use of everyday tools. It is interesting, however, to reflect about how the progressive generality of machines around us will lead to an uncharted territory in the theory of design, where humans will no longer be able to predict the behaviour of machines, but start treating them closer to unpredictable living things. The first computers were used by few experts worldwide, and now smartphones are widespread and allow users to talk with powerful assistants often indistinguishable from actual humans. Similarly, in robotics, industrial robots are designed to be programmed by experts and operate in special environments. Evolution in technology led to robot vacuums, more common in the home as they are of intuitive use, whose function and limits are clear, but whose behaviour is nevertheless less predictable. The evolution in cognitive and dexterous capabilities of robots, together with a likely anthropomorphic aspect, might blur the mental map of what users can expect a robot to do, incapable of recognising clear structural limitations as their form appears closer and closer to humans.

[3]We mention *pipelines* as the collection and generation of data and the efficient deployment of such models, while seen often as an engineering problem, is of crucial importance for the field.



Figure 1.1: The behaviour of machines around us has evolved over the last decades, and the rapid improvement of learning algorithms has modelled it to appear closer to human behaviour. At the same time, these machines face increasingly unpredictable environments, and can respond with equally unpredictable behaviour. Images taken from computerhistory.org, covariant.ai/robotic-induction/, blog.google, xiaomi.com, zdnet.com, ami.iit.it/human-robot-collaboration.

of data. The Web is a great source of data for computer vision or natural language processing algorithms. Fleets of cars driven around by people are collecting millions of hours of examples of how to drive, that can then be distilled into a machine learning model. For robotics, we do not have any large enough data set sitting in some server farm, nor do we have robots being used to collect data at a reasonable scale.

This is the main problem studied in this thesis and the main contribution of all the papers that make up it: *addressing the need for data from learning algorithms.*

The first paradigm we establish is self-supervised data collection for manipulation. We demonstrate that robots can autonomously generate the vast majority of training data they need by decomposing tasks into alignment and interaction phases. Our key insight is that by learning to align the end-effector to a "bottleneck pose" relative to objects, robots can replay recorded interaction trajectories without needing to learn the fine-grained dynamics of object manipulation. This approach, developed progressively from single-task learning (Chapter 4) to multi-object generalisation through retrieval (Chapter 5), reduces human demonstration requirements by 10× compared to standard behavioural cloning while maintaining or improving task success rates. The second paradigm we establish is leveraging foundation models for robot

manipulation without robotics-specific training. We show that vision and language models, despite being trained on non-robotics data (web images and text), encode sufficient geometric and semantic understanding to enable efficient robot learning. Our progression from using features extracted from foundation models for visual servoing (Chapter 6) to repurposing large language models as "imitation learning machines" through our Keypoint Action Tokens framework (Chapter 7) demonstrates that the capabilities of these models can be directly translated into robot performance improvements—without any robotics fine-tuning. This finding has profound implications: as foundation models continue to improve through investment in other domains, robot manipulation capabilities can improve in tandem. The culmination of these ideas (Chapter 8) demonstrates that robots can learn manipulation skills directly from unstructured human videos by combining retrieval with in-context imitation learning. This points toward a future where robots continuously learn from the vast and growing corpus of human demonstration videos available on the internet, dramatically lowering the barrier to teaching robots new skills.

Specifically, these are the contributions described in each technical chapter of this thesis:

- Chapter 4 introduces Self-Replay, a technique that allows robots to autonomously collect data needed to solve long-horizon, multi-stage tasks from a single human demonstration (published at the *Conference on Robot Learning (CoRL) 2021*).

  Original paper title: *Learning Multi-Stage Tasks with One Demonstration via Self-Replay*, Authors: *Norman Di Palo, Edward Johns.* Contributions: *The project was led by Norman Di Palo, including design and development of methods, experiments, and writing, under the supervision of Edward Johns.*

- Chapter 5 extends the previous method, designed to learn tasks in isolation, into a framework that can ingest increasing amounts of data and tasks over time, and result in more general behaviour through the use of retrieval from the past experience of the robot (published in the *Robotics and Automation Letters (RA-L) 2024*).

  Original paper title: *On the effectiveness of retrieval, alignment, and replay in manipulation*, Authors: *Norman Di Palo, Edward Johns.* Contributions: *The project was led by Norman Di Palo, including design and development of methods, experiments, and writing, under the supervision of Edward Johns.*

- Chapter 6 introduces DINOBot, a method that builds on the capabilities of recent vision models, and designs robot behaviour around their strengths. Remarkably, we show that

the ever-improving capabilities of vision models, trained on increasingly large datasets of images scraped from the web, can be directly translated into improved robot performance, without the need for robot data (published at the *International Conference on Robotics and Automation (ICRA) 2024).*

Original paper title: *DINOBot: Robot Manipulation via Retrieval and Alignment with Vision Foundation Models*, Authors: *Norman Di Palo, Edward Johns.* Contributions: *The project was led by Norman Di Palo, including design and development of methods, experiments, and writing, under the supervision of Edward Johns.*

- Chapter 7 introduces Keypoint Action Tokens, a technique that uses Large Language Models as the backbone to imitate human behaviour. We demonstrate that, by translating observations and actions into a "language" that such models can understand, we can unlock few-shot, in-context imitation learning on a large series of everyday tasks (published at the *Robotics: Science and Systems (RSS) conference 2024*).

  Original paper title: *Keypoint Action Tokens Enable In-Context Imitation Learning in Robotics*, Authors: *Norman Di Palo, Edward Johns.* Contributions: *The project was led by Norman Di Palo, including design and development of methods, experiments, and writing, under the supervision of Edward Johns.*

- Chapter 8 extends Keypoint Action Tokens by adding the capability to retrieve clips of human executing tasks, conditioned on a desired command, from long, uninterrupted videos of people performing tasks in their natural environments. We call this method Retrieval + Execution, and demonstrate that it can perform in-context, human-to-robot imitation learning from an ever-increasing dataset of human videos, without the need for retraining (published at the *International Conference on Robotics and Automation (ICRA) 2025).*

  Original paper title: *R+X: Retrieval and execution from everyday human videos*, Authors: *Georgios Papagiannis*, Norman Di Palo*, Pietro Vitiello, Edward Johns. * indicates equal contribution.* Contributions: *The project was jointly led by Georgios Papagiannis and Norman Di Palo, including design and development of methods, experiments, and writing, under the supervision of Edward Johns. Pietro Vitiello supported the devlopment of the baselines.*

Data and large models have been and will be instrumental to the growth of capabilities in robots, and we believe the contributions made in this thesis are of clear relevance to tackle the

main challenges the field is facing, from generating the data to understanding how to best use the existing powerful, but not robotics-focused models the machine learning community has developed.

The next chapter will provide a deeper analysis of the recent history of AI and robotics, elucidating how the current techniques emerged as a promising solution to tackle general robot learning. It also describes the impact robot learning can have on society, and why it is a fundamental area of research that is attracting the world's attention.

Chapter 2

Society and Robots: The Ideas and Events that Shaped the Field

> *The Zeroth Law*: "A robot may not harm humanity, or, by inaction, allow humanity to come to harm."
>
> Isaac Asimov

2.1 Part 1: What to Optimise and Meta Problems

Why would we want to develop robots that can act in everyday human environments? We will describe in the next paragraph why this problem is of great importance, and in continuation to what humanity has been working on for most of its history. To understand this, we need to take a step back and look at the big picture.

2.1.1 The Big Picture

Throughout history, humans have always worked to improve their condition and well-being. This has sometimes resulted in benefits to society as a whole. However, condition and well-being are generally subjective terms. Despite this, by reasoning in terms of first principles, we can identify a set of metrics, rooted in physical and mathematical properties, that correlate to these terms.

In general, the state of a society can be expressed, at least partially, as their ability to generate goods, such as food, energy, and housing, and services, like healthcare, education, transport, and entertainment. The abundance and ability to generate such goods and services

results in the ability of each member of society to access them. The way these goods and services are distributed across society is beyond the scope of this work, but we will assume that abundance will lead to horizontal distribution. [14, 159, 184]

From a big picture perspective, we can formulate the generation of such goods and services as requiring three main inputs: *energy*, measured in Joules (J), *time* spent by people, measured in seconds (s), and the number of people assigned to the task as a positive integer P [186]. We can therefore say that a desirable metric to optimise is the ratio between good and services, globally described as desirable output O_{des}, and the set of energy, seconds and number of people required, globally referred to as necessary input I_{nec}: $c \propto O_{des}/I_{nec}$.[1].

This ratio is defined, generally, by the state of technology of a certain society. This refers to the set of knowledge, techniques, and machinery that allow to obtain a certain output given a certain amount of energy J, time s, and workers P. The invention, design and deployment of technological advancements is what allows to reduce the needed input I_{nec} and therefore maximise the aforementioned ratio c [7].

Inventions and technological advancements stem from research, that can be seen as an investment of I_{nec}: instead of utilising this input to generate O_{des}, it is allocate to the advancement of technology [7]. This leads to a momentary decrease in the aforementioned output, that is however compensated by a permanent improvement in the ratio c once a breakthrough is obtained.

There is, in addition, another distinction in the form of technological advancement and kind of problem that is addressed. Some solution address the problem itself. For example, quern-stones are tools that allow people to grind grain. This tool tackles directly the desired service of grinding grain (resulting in a new desirable good), allowing fewer people and less time and energy needed to complete the same tasks. There is, however, the possibility of solving a *meta problem*. We refer to *solving the meta problem* as the design of a solution and machinery that does not help in a single instance of a task, but can tackle the majority of such tasks in a fundamentally more efficient way. A meta solution for grinding grain is a mill. The mill, once deployed, drastically reduces the inputs J, s, P, allowing the semi-autonomous completion of the grinding task, the improvement of the ratio c, and the access to the desired good *ground*

[1]This definition of the desired outputs and needed inputs is generally aligned with definitions that have been given in the field of economics and political sciences. However, the main inputs of productions are generally defined as *labor, capital, land* [12]. These are strongly correlated to the values we use, that are however more physically grounded. Furthermore, we abstract away the concept of capital and can consider the moving of capital ininfluent when considering all of humanity as a closed system, ignoring concepts such as profit and incentives to focus on the metrics we mention above. Energy and time are the only physically measurable values that are effectively **spent** in a non reversible process. [14]

grain to a vaster population [7].

The drawback of solutions to meta problems lies almost entirely in their difficulty and need for a large investment of J, s, P before a solution can be deployed at scale and lead to a return. Therefore, for a long time, most problems have only received direct solutions that partially optimised their production, instead of meta solutions.

2.1.2 Meta Solutions and the Rise of Automation

The effectiveness, and indirectly also the definition, of a meta solution lies in its ability to perform a task *autonomously*. In the example before, the difference between a mill and a quern-stone is the former's ability to perform the task with close to no human intervention. It is a clear example of automation, a property of certain solutions that unlock a substantial improvement in maximising c by reducing the need for P. While giving an exact definition of a meta solution is challenging, we can consider it as any advancement in the technological knowledge of a society that requires a substantial effort in J, s, P to be achieved, but provides a return that is considerably larger than any other more direct solutions [223].

The exponential effect of meta solutions and automation on society can be seen in the Industrial Revolution [6], that eventually led to an unprecedented increase in the production of certain goods. Likewise, the advent of computers, the Internet, and the age of information led to the same problem-solving patterns being applied to a vast set of services, now fundamentally more accessible to the general population (e.g. access to the world's information, entertainment, government services). In this work, we will focus on physical tasks, revolving on the manipulation and interaction with objects, tools, and environments.

The research and development process that led to the deployment of automated solutions for the efficient generation of goods and services has followed the same pattern for large part of humanity's history: in the beginning, humans allocated their time, abilities and knowledge to perform a task resulting in the aforementioned desired output. Over time, tools have been developed to reduce the amount of time, energy and people needed. Then, once the technological background reached a critical maturity, a meta solution was researched to solve certain problems with a dramatic reduction in these quantities [223]. Other examples include the passage from efficient hand-writing to the printing press, from manual cloth washing to the washing machine, and so on.

In industrial settings, where goods are designed and produced, this led to the adoption

of robots[2]. Again, designing machines that can automate any process in a production and supply chain has proven to be dramatically effective in reducing the need for I_{nec}. Over the decades, therefore, generations of engineers have focused on investing their time and energy into these research efforts, obtaining meta solutions whose return has been greatly larger than the investment.

This has been, for most of the last couple of centuries, the way automation, and robots, have progressed, becoming more and more pervasive in the functioning of society, albeit they may operate hidden from most people's eyes. The history of success of this pattern of problem-solving therefore led engineers, researchers and designers to try and apply it to more and more physical tasks, in the hope to automate away every boring and mundane task of our life. Robots have been reliably building cars, rockets, and nanometer scale processors, how hard could it be to create a robot that can tidy up offices?

Surprisingly hard. The reason for this lies in the failure of finding a meta solution for what became an intractably large number of meta problems.

2.1.3 Too Many Meta Problems: the Long Tail of the Mundane

Think about how many physical interactions you have with your environment every single hour. If we ignore mobility and just focus on manipulation, a lot of objects get interacted with in remarkably different ways to obtain many different goals. Opening a fridge by first grasping its handle and pulling it, to localise and grasp a jar of sauce, picking it and removing it from the fridge while closing it, opening it by twisting its lid and then pouring it on a dish that required an enormity of interactions to be prepared. This is fundamentally different from grinding grain. Not only that, it is fundamentally different from producing goods in a factory, that specialised on a few of them and has an entire environment designed for that process. Humans, on the other hand, adapt to any new environment[3]. Most importantly, humans perform an enormity of tasks in their environments, that millions of years of evolution made simple and intuitive, and led to a form of blindness to their inherent difficulty. The act of opening a fridge reliably,

[2]A comprehensive definition of what a robot is is surprisingly evasive. One possible definition is *any automatically operated machine that replaces human effort, though it may not resemble human beings in appearance or perform functions in a human-like manner.* This would make a washing machine a robot. Another definition, more commonly used in academic settings, is *a robot is a machine able to intelligently map sensorial inputs to actions.* This leads to the problem of defining what *intelligence* means, a possibly even more challenging task, and the fact that most industrial robots would not be, by this definition, robots, as they can act almost blindly and react in a generally not-so-intelligent way to any unexpected circumstance.

[3]While it can be argued that your house is designed by you and for you within a certain range, many everyday environments offer less care in their design, requiring a non-trivial mental and physical effort to be navigated and interacted with.

for a robot, would necessitate a meta solution. Likewise localising a desired jar, and so on.

The cycle of finding meta solutions to meta problems is therefore dwarfed by the long tail of apparently easy mundane task that any robot would need to do to be considered generally useful in such environments. The time and energy spent on designing a solution to these tasks that would work in any new household or office would no longer lead to a return in terms of optimising c: therefore, the design and deployment of such solutions have been, after an initial period of optimism, generally discarded, leading human environments to humans. No machine could deal with such scenarios as they would require an unprecedented form of *intelligence*, that we here define as the ability to quickly adapt to new situations, gain knowledge from experience, and effectively map sensorial inputs to physical actions.

What if the same problem-solving pattern was applied in an even more abstract way? A plethora of single tasks faced a great improvement in efficiency by tackling the meta problem with a meta solution. Reasoning by induction, this suggests that to tackle a plethora of meta problems, we need a meta meta solution or meta^2 solution.

2.1.4 Abstractions All The Way Down: Intelligence as the Meta^2 Solution

Following the reasoning proposed before, a meta^2 solution would be any technological advancement that can automatise the solution to meta problems. A machine, not intended as a physical being but as a mapping from inputs to outputs, that can obtain meta solutions while requiring substantially less time and effort. This could potentially solve the efficiency issue regarding the amount of time, energy, and people needed to reach such a solution and the return of investments in the form of a drastic reduction in the need of such inputs to obtain the same desired outputs. From a purely theoretical point of view, this second layer of abstraction could lead to a breakthrough optimisation of c, but under one circumstance: **the amount of inputs J, s, P needed to arrive to such a meta^2 solution must be exponentially smaller than the sum of such quantities needed to find meta solutions to all the meta problems the meta^2 solution can solve**.

This therefore leads to a paradigm shift with respect to the pattern we mentioned before, especially if we consider the field of robotics. The inability to tackle mundane tasks as we had tackled industrial tasks led the research field to study this problem from a broader perspective. As robots needed to enter human environments and mirror more closely the behaviour of humans, the idea emerged that robots needed to be more *intelligent*. Without the need for

a formal definition, the ability that a machine able to interpret a plethora of inputs, such as visual and tactile, and convert it into sensible actions that lead to a desired goal can be seen as (at least in a certain measure) intelligent seemed intuitive. This seems intuitive especially when intelligent is defined in contrast to industrial robots. The latter operate in engineered environments and execute the same manipulation actions over and over, as the designers assumed the production chain itself minimised any possible form of uncertainty, that could have been further removed with the use of trivial sensor-based algorithms. Intelligent robots, however, would need to be able to autonomously operate in unstructured, human environments, and perform a large set of tasks humans generally perform. As we consider ourselves intelligent, the logical implication seemed correct, at least at a first approximation.

Given these observations, the fields of robotics and artificial intelligence started to overlap. The pursuit of an algorithm able to mimic, at least partially, human intelligence, resonated with robotics researchers looking to empower robots with skills that were challenging to manually design, especially at the required scale. At the same time, this made robotics share the successes and failures of artificial intelligence from the past few decades.

The next part of this chapter will lead the reader along a quick tour of the recent history of AI and robotics, and the major events that shaped the fields.

2.2 Part 2: A Brief (and Shared) History of Robotics and Artificial Intelligence

The fields of robotics and artificial intelligence have long been intertwined, each influencing and driving the other forward. Their shared history is marked by periods of rapid advancement, paradigm shifts, and occasional setbacks. To understand the current state of robotics and AI, and the motivations behind this thesis, it is crucial to examine some key milestones in their development.

While the history of AI and robotics spans many decades, we will here put the spotlight on a few defining moments that strongly shaped the following years, or decades, of research. Understanding why these moments were crucial, and how they influenced the field, is paramount to better understand the current state of technology, and the set of techniques and tools that are explored in this thesis as a possibility for advancing the field.

2.2.1 Experts and Search: the First AI Wave

In 1997, the world watched as IBM's DeepBlue chess computer defeated the reigning world champion, Garry Kasparov. This event marked a significant milestone in the history of AI, demonstrating that machines could outperform humans in specific, well-defined tasks that were once thought to require human-level intelligence. Few believed this achievement was possible, so much so that many assumed it was a set of chess grandmasters that were maneuvering DeepBlue under the hood, a doubt that even Kasparov raised. Deep Blue's architecture was emblematic of the AI approaches prevalent at the time: a combination of brute-force search algorithms and expert systems. This approach relied heavily on processing power to evaluate millions of possible moves, and approach generally defined as *search*, coupled with handcrafted rules derived from chess grandmasters' expertise.

Chess possesses many of the features needed to use search in order to plan the next move. First, it is a perfect information game: all players can observe the state of the game in its entirety (the chessboard configuration), and there is therefore no uncertainty over the current state. As a counterexample, many card games, such as poker, are not perfect information, as a player does not know the cards in the opponents' hand, and needs to plan taking this uncertainty into consideration. Secondly, chess is an entirely deterministic game: when selecting a move, the outcome is entirely predictable. Therefore, there is no need to plan taking into account the distribution of possible futures influenced by the uncertainty of the dynamics of

the game. Another counterexample to this is any game that involve throwing dices or other stochastic elements: in such games, a player must plan taking into consideration these sources of randomness that can alter the outcome of a move.

Given these properties, it would seem that a computer would only need to search over all possible future configurations of the board to find the move that would lead to the optimal one. This is over only partially true. Chess, despite possessing the above features, has also a series of inherent difficulties for a search algorithm. First, its branching factor (the number of different move that can be chosen at any turn, on average), and average game length, lead to a vast amount of moves and possible chessboards configurations to explore. This leads to the need for considerable compute power.

The second is the difficulty to evaluate most chessboard states: while there are well-defined rules to define a victory, a defeat, or a draw, there are no way to numerically evaluate intermediate chessboard configurations. Over centuries, chess experts were able to identify a "value" for each piece, and find a set of configurations from which a victory is almost always assured. By manually providing these insights to the algorithm, the search process can be guided further. This is a perfect example of an "expert system" that is informed by a set of rules and facts from experts of the field.

The success of Deep Blue in chess mirrored similar approaches in robotics, where planning and expert systems were seen as key to achieving intelligent behaviour. These systems often relied on first-order logic and symbolic reasoning to represent and manipulate knowledge about the world. A pioneering example was the robot "Shakey," developed at the Artificial Intelligence Center of Stanford Research Institute in the late 1960s, which used logical reasoning to plan and execute actions in a simple environment. Although Shakey predated DeepBlue, it exemplified the belief that scaling up such systems could lead to general-purpose robotic intelligence.

Shakey was a wheeled robot with no arms of any kind. It could perceive the environment through a series of sensors, including a camera. It could navigate corridors to reach different rooms, climb on top of small obstacles, and push cubes. In order to use these aforementioned AI techniques, the designer had to "translate" the real world in a representation not too dissimilar from a chessboard. The environment was represented as a series of states, with Shakey able to observe its environment through a camera and compute, at least partially, its state and the state of the environment. It could then select a series of actions, mostly composed of movement to goal positions. Using a model of the environment, it could plan a series of actions that would lead to the desired state, the latter being often a command received by a human operator. The similarities with a chess-playing AI are, therefore, clear.

There are, however, a series of differences that makes the control of a robot fundamentally more difficult than controlling a chessboard. First, we mentioned the states. A chessboard has, however large, a finite number of possible states. These are also entirely observable, as we specified when talking about observability. The physical world cannot be reduced to a set of finite states, and is also far from being perfectably observable. At any given moment, we have a rough estimate of the state of our surroundings, but we ignore most of the state of our environments, even considering things as simple as rooms behind closed doors. The sensorial inputs are also very high-dimensional and difficult to interpret: Shakey relied on simple computer vision techniques to find the edges of walls and objects, in order to have an estimate of its surroundings. In the physical world, a robot would need to understand and analyse a continuous, high-dimensional stream of images, sounds, and tactile information to mimic what we perceive as humans. Furthermore, the robot possessed a series of limited actions that would then be converted into low-level control for its motors. In the physical world, the possible actions to execute are extremely nuanced and fine-grained. Even when finding a good set of "abstract" actions, like moving to a room, or picking up an object, understanding how to physically execute such an intermediate plan is not trivial.

It is clear, therefore, that these techniques that would eventually "solve" chess could not be trivially transported to the physical realm of robotics. Despite this, the Shakey project led to a plethora of fundamental findings in the fields of planning, perception, and more, like the A* search algorithm, a fundamental algorithm for finding shortest paths in graphs, or the Hough transform, a technique to extract features from images. The project pushed the boundaries of robotics planning languages in the following years.

This approach culminated in the development of the Planning Domain Definition Language (PDDL) in 1998, which provided a standardised format for describing planning problems and domains. PDDL and its variants became widely used in automated planning and scheduling, reinforcing the idea that complex robotic behaviours could be achieved through increasingly sophisticated symbolic reasoning and planning algorithms. Many researchers believed that by further refining these techniques and increasing computational power, the field could eventually produce robots capable of human-like reasoning and adaptability across a wide range of tasks.

However, this approach failed to meet expectations. The real, physical world is too nuanced and has a long-tail of possibilities that is virtually impossible to manually list, describe and link together in a graph. In addition, the increasing size of possible entities and their relations made search algorithms quickly impractical, as the branching factor increased exponentially. The ability to formulate a plan was also deeply rooted in the robot's ability to accurately



Figure 2.1: Two pivotal moments for AI involved a boardgame match against a top ranked professional player. The victories of the AI systems were unprecedented, but for two different reasons, as we describe in this chapter. Figures taken from independent.co.uk, newyorker.com.

determine the state of the environment through raw sensory inputs, such as vision. This proved to be extremely challenging, even for tasks that would seem quite trivial to a human. Furthermore, the ability of a robot to formulate a high-level plan does not coincide with the ability to execute it, as we mentioned before. Robots like Shakey could indeed navigate known, simple, controlled environments, and manipulate a few well-defined objects, but were never able to reliably perform tasks outside of a lab setting. It quickly became clear that machines lacked a fundamental aspect of human intelligence: intuition that went beyond formal logic.

2.2.2 Intuition through Experience: the Rise of Reinforcement Learning

Nearly two decades after DeepBlue's victory, another AI milestone captured the world's attention. In 2016, DeepMind's AlphaGo defeated Lee Sedol, one of the world's top Go players, in a game long considered too complex for machines to master.

From a high-level perspective, Go resembles quite closely chess, especially in light of the properties we listed before, which made chess a good testbed for search algorithms. Why was this therefore considered another breakthrough? The reason lies in the branching factor of Go. Unlike chess, Go's vast number of possible moves at each turn and board configurations made pure brute-force approaches infeasible. At the time, the breakthrough was considered still a couple of decades away, as the predictions simply extrapolated the delta of the increase in computational power of super computers, still imagining brute-force search and expert systems as the only way to tackle the game.

AlphaGo's success stemmed from a fundamentally different approach. Instead of relying solely on pre-programmed rules and exhaustive search, AlphaGo utilised deep neural networks trained through reinforcement learning. In simulation, the algorithm played extensively against itself, emulating thousands of years worth of matches and receiving positive or negative rewards when a match was won or lost. It explored an enormous variety of possible moves in many configurations of the board via random search, and then evaluating the domino effect they had on the eventual outcome of the game. By allocating TFLOPs (Tera FLoating point Operations Per second) of compute during training to simulate millions of matches, reinforcement learning allowed to eventually distill this knowledge in a few neural networks that were able to recognise patterns on the board to evaluate the probability of winning from a certain configuration, and only a handful best possible next moves.

More concretely, the experience gathered by AlphaGo was distilled into two neural networks: a policy network, and a value network. The former was used to select actions given a board configuration. This allowed to dramatically reduce the branching factor of Go: just as a human expert player would, the policy network proposed a few moves that were considered the most promising, avoiding using expensive but useless research in trivially suboptimal moves. The latter, the value network, was used to provide a "score" for each configuration of the board. Intuitively, it would predict the probability that each player would win given a board configuration. This would allow to avoid searching for too many moves in the future. As we described before, these games have clear winning conditions, but are difficult to evaluate mid-game. Although DeepBlue relied on a set of hand-crafted rules and desired configurations to reach, AlphaGo was able to learn to score the board configuration autonomously, through its vast experience playing the game[4].

This allowed the system to develop its own intuition about the game, learning strategies that sometimes appeared alien even to expert human players. This form of emerged intuition, which can be seen as learnt pattern recognition, made tree-search algorithms tractable again, and much more effective. The combination of learnt pattern recognition and search led to a set of surprisingly but eventually winning moves that took the world by surprise. The triumph of AlphaGo represented a paradigm shift in AI. It demonstrated that machines could not only process vast amounts of data but also develop their own strategies and insights. This success

[4]It is worth noting how the first version of AlphaGo, the one that eventually led to the win over Lee Sedol, started its training by imitating hundreds of thousands of expert human matches, and then fine-tuned its abilities by self-play. The method was then extended to rely entirely on self-play and start with zero human data, leading to a new method named AlphaGo Zero that eventually surpassed AlphaGo, demonstrating how, ironically, human data was almost a bottleneck.

sparked renewed interest in neural networks and reinforcement learning across various domains, including robotics.

Similarly, around the same time, DeepMind demonstrated how a single reinforcement learning algorithm, without any change in architecture or hyperparameters, was able to master a wide range of Atari videogames, from pixels alone. This further sparked interest in scaling up reinforcement learning algorithms: while Go was a single, extremely challenging game, the Atari project showed that the same algorithms could learn to play a range of games without any prior knowledge.

In robotics, this shift manifested itself in attempts to apply reinforcement learning to complex control tasks. Researchers hoped that robots could learn to navigate complex environments and perform intricate manipulations through trial and error, much like AlphaGo learnt to play Go or as AI agents were able to beat Atari games through experience alone. This approach promised to overcome the limitations of hand-engineered rules, allowing robots to adapt to uncertain and dynamic environments. However, despite some impressive demonstrations in controlled settings, the application of reinforcement learning to real-world robotics tasks proved challenging. Both Go and Atari agents were trained entirely in simulation: therefore, they were not constrained by the speed of physics, and could also reset the environment at will. This is obviously not possible in the real world. To bridge this gap, much research was dedicated to developing physics simulators that could simulate the real world as accurately as possible. This approach, which is still actively researched, has brought about a series of strong results, but until now fell short of the possibly overly optimistic expectations to trivially transfer such algorithms to the robotics realm. The challenges are many: a simulator may not resemble perfectly the real world both in terms of visual fidelity, and physical properties, such as frictions, contacts, etc. Finally, robotics needs to operate in much more open-ended and varied environments, while Go and the Atari videogames are self-contained.

Reinforcement learning had enormous impact on many areas of computer science, and even in robotics, but was certainly not a trivial solution to general, human-like robots. The field is particularly active, and breakthroughs in neighbouring areas of AI often influence each other, with results that were originally developed for game-playing agents eventually leading to impressive improvements also in the fields of natural language processing, science, mathematics, and computer vision.

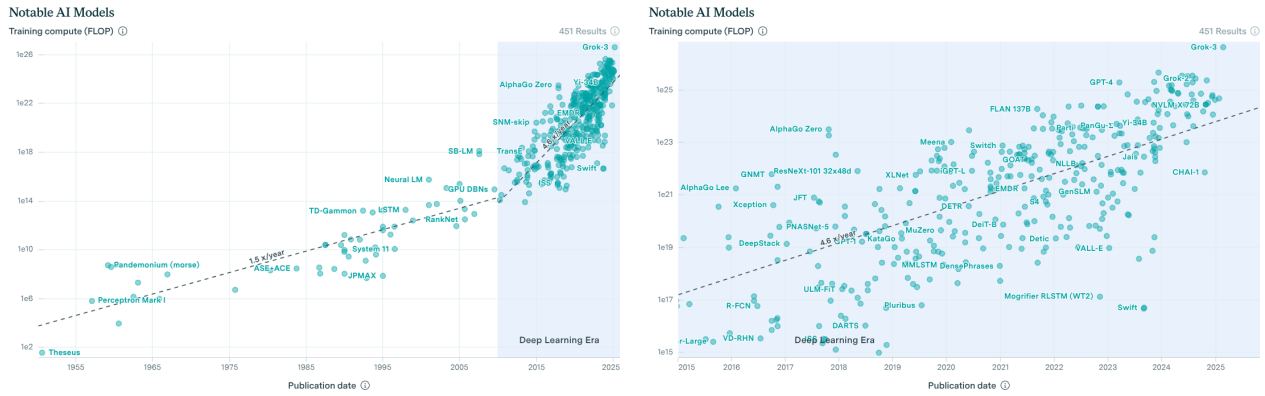


Figure 2.2: These figures show the increase in computing power, measured as FLOPs, used on notable models over the decades. The right figure focuses on the models released after 2015. Figures taken from https://epoch.ai/blog/compute-trends.

2.2.3 The Unreasonable Success of Scaling: Intelligence from Data

The dawn of the 2010s was characterised by a series of pivotal moments for AI, in particular, what will then become the field of deep learning (DL). One fundamental moment was the enormous success of deep Convolutional Neural Networks (CNNs) in visual recognition tasks. AlexNet, a CNN developed in 2012 by Alex Krizhevsky, Ilya Sutskever, and Geoffrey Hinton, won the ImageNet classification competition, clearly surpassing its competitors. The competition measured the ability of computer vision algorithms to receive an unseen image and classify it into one of a set of possible labels that ranged from animals, vehicles, and more.

Most of the leading algorithms, at the time, were based on hand-designed algorithms to extract features and patterns from images, and often using "shallow" machine learning techniques on top of these features to classify these images. The paradigm shift introduced by AlexNet was to demonstrate how a deep neural network could ingest raw pixels and output labels without the need to manually design feature extractors. The network could learn to autonomously design feature extractors using pure supervised learning: it received thousands of examples of *(image, label* and adapted its weights through gradient descent. Parallel computation, like the use of GPUs, was developed by Krizhevsky to dramatically sped up the learning process of these techniques, a methodology that would quickly become the backbone of modern AI.

AlexNet drastically reduced the error rate on the ImageNet challenge, demonstrating that the real fuel to improve AI systems was not better and better algorithms, but more and more data and computing power, a recipe that will stand the test of time over and over again (at least in the decade that went between that moment and the writing of this thesis).

The success of AlexNet was not an isolated event, but rather the beginning of a trend that would come to define the field. Subsequent models such as VGGNet, GoogLeNet, and ResNet

continued to push the boundaries of performance on computer vision tasks. The novelty in architectural design was mostly proposed to achieve a common goal: to make these networks deeper and deeper without altering their abilities to learn. A common thread emerged: the power of scale. The research found that increasing both the size of the training dataset and the complexity, or size, of the models led to consistent improvements in performance. These results and patterns were not limited to computer vision: similar breakthroughs were obtained in natural language processing and other fields of computer science where data is abundant. The mantra "more data, bigger models" (Fig. 2.2) became a driving force in AI research, with results consistently showing that scaling up often led to qualitative leaps in capability.

This trend continued to obtain successes year after year, with research focus shifting from finding the best architectural design, to obtaining more data to train on, to finding more and more compute power and efficient training techniques. A fundamental breakthrough came in 2017 with the design of the Transformer. Originally designed for machine translation, Transformers quickly proved to be an extremely versatile and powerful architecture for a wide range of tasks, spanning from computer vision to natural language processing and even robotics. Transformers, in a certain sense, were another step towards the direction of removing all human-designed inductive biases, and allowing a very general architecture to learn any possible patterns in data autonomously. CNNs were powered by convolutional layers, that were based on the idea that visual patterns would often be local and could appear in any part of the image. Transformers remove this bias, which still allowed networks to learn from datasets of reduced dimension. Thanks to their core self-attention mechanism, Transformers autonomously learn to find patterns between any parts of the input, without being constricted by locality hypothesis, like in the case of CNNs, or by the assumption that list of inputs, like time-series or natural language sentences, should be read in temporal order, as it happens with Recurrent Neural Networks.

The exponential growth of this trend over the last decade led to the development of what were named Foundation Models: networks trained on enormous amounts of data with previously unimaginable amount of TFLOPs. While these networks possesses extraordinary capabilities in many tasks, like computer vision and natural language tasks, their great advantage is the ability to be quickly fine-tuned for new tasks using a small amount of data, and effectively transfer all the knowledge that was distilled from web-scaled datasets. Signs of groundbreaking results in the field of natural language processing began to emerge with the release of GPT-2, the first language model capable of creating coherent paragraphs of text. Simply scaling this architecture to orders of magnitude more data led to unprecedented results that eventually

captured the world's attention with the advent of ChatGPT, a language model finetuned for chat and instruction-following and based on the GPT-3 model.

The general language abilities of these language models, including their apparent common knowledge of the world, reasoning abilities, and the ability to follow instruction, were unimaginable even a few years before their release. These results instilled the idea that, if such an intricate task as language could be learnable from data alone, then everything else could, including robotics.

The advent of foundation models and large language models has begun to revolutionise the field of robotics. These models, with their ability to understand and generate natural language, are being leveraged to create more intuitive interfaces between humans and robots. For instance, researchers have demonstrated robots that can understand complex, multi-step instructions given in natural language, translating them into actionable sequences of movements and manipulations. Moreover, the general knowledge encoded in these models is being used to enhance robots' understanding of the world around them. By combining language models with computer vision systems, robots can now identify and reason about objects in their environment with unprecedented sophistication. Perhaps most importantly, the few-shot learning capabilities of large language models are being adapted to robotics, allowing robots to quickly learn new tasks from just a few examples or descriptions. This marks a significant step towards more adaptable and general-purpose robotic systems, capable of operating in diverse and dynamic real-world environments.

The years following the release of these models saw two trends in robotics: global research efforts to understand how to best use these models to improve the abilities of robots, as just described, but also efforts in understanding how to develop robotics models of similar sizes and capabilities. This thesis will focus on both of these aspects, with a particular emphasis on the former. However, we want to note how our lab participated in the largest (at the time of writing) global effort on building robotics foundation models by first gathering an enormous dataset of robotics tasks from institutions all over the world and then training a single large model on these. This effort results in the publication of the Open X-Embodiment Dataset and Models, whose paper won the 2024 Best Paper Award at the International Conference of Robotics and Automation (ICRA). Although the size of language or computer vision datasets still dwarfs the size of these robotics datasets, this global effort demonstrated that such a path is possible and contributed to the vision that a shared effort in data collection can lead to unprecedented abilities in robotics.

The "unreasonable effectiveness" of scaling in AI and robotics has fundamentally altered

the trajectory of these fields. What began as a trend in computer vision has evolved into a paradigm that spans across domains, from natural language processing to robotic control. The success of this approach has challenged long-held assumptions about the nature of intelligence and learning, suggesting that many cognitive capabilities can emerge from statistical patterns in large datasets, given sufficiently powerful models to capture them.

However, despite the global efforts in collecting robotics data, reaching the amount of data that theory suggests will enable a similar breakthrough is still not trivial or within close reach. The hunger for data of many recent DL techniques, justified by their enormous success, requires deeper thinking over how to best collect robotics data and transfer and reuse more widely available non-robotics data.

This is the main motivation behind this thesis: an effort in designing techniques that can **both generate robotics data with little effort**, and reduce the need for the former by **skilfully using data and models from other domains**, leading to algorithms that enable robots to learn new skills efficiently. This vision is aligned with the fact that robotics is, in its quest to model the full spectrum of human intellectual and physical capabilities, a fundamentally multimodal discipline. In particular, chapters 4 and 5 will present techniques we developed that enable robots to autonomously collect the data needed to train machine learning models that can tackle a series of everyday tasks. Chapters 6 and 7 introduce a different approach to efficient robot learning, where instead of making collection of robotic data more efficient, we demonstrate how robots can benefit from the enormous datasets of images, and even text, that are available on the Web, even if they are not directly tied to robotics. Chapter 8 focuses on extending these ideas to allow robots to passively learn from watching humans perform tasks in their natural environments, a path that would also unlock an unprecedented amount of data.

Chapter 3

A Gentle Introduction to the Ideas of Robot Learning: Theoretical and Literature Background

Does the body rule the mind
Or does the mind rule the body?
I don't know.

Still Ill - The Smiths

In this thesis, we focus entirely on robotic manipulation. Robotic manipulation studies the interaction of robot arms with their environment, composed of everyday objects and scenarios, with the intent of enabling the robot to execute a range of everyday activities. In particular, "*the study of robotics concerns itself with the desire to synthesise some aspects of human function*", as described in [41].

In order to be effective in natural environments, a generalist robot should be able to *see*, to *act*, to *understand commands*, and to *think*, or more specifically, to connect vision and commands to actions. In this chapter, we give a gentle introduction to the above aspects, describing the main concepts behind such abilities, and how we will use them later in this thesis.

We will then focus more on the recent robot learning literature, focused on the link between perception and actions, and how different techniques and seminal works both shaped the field and inspired the works presented in this manuscript.

3.1 Acting: A Brief Introduction to the Kinematics of Robot Arms

A robot arm is composed of a chain of links, that are rigid bodies, and joints that connect them and allow for movement. At the end of this chain, we find an end-effector. The end-effector can be of many kinds, designed around the task to be executed by the robot. In this thesis, we will always assume that the end-effector is a parallel-jaw, two-fingered gripper. [1] Through the use of joints and the end-effector, the robot arm can move in a *workspace*, that is the area reachable by its end-effector, and manipulate objects and the environments. The tasks executed by the robot can therefore be described, at their core, as the sequence of poses of the links of the end-effector over time, the forces they apply on the environment, and the sequence of poses of any object of interest over time.

The study of the movements of the robot arm, the forces applied by it on the environment, and the movement of the manipulated object is strongly rooted in the study of mechanics, and the relative tools from mathematics, such as linear algebra and calculus. In particular, we can study the movement of a robot arm through the lens of its kinematics and its dynamics. The former studies movement, without regard for the forces and torques that generate such movements. The latter, instead, focuses on the study of the differential equations linking the forces and torques applied by joints and the environment and how they translate into accelerations.

Kinematics can also be divided into forward and inverse kinematics. Forward kinematics is the analysis and synthesis of a function that maps the configuration of the robot's joints and outputs the pose of a link of interest on the robot arm, very often the end-effector. It can therefore be expressed as a function $x = f(\theta)$, where x represents the end-effector pose (position and orientation) in Cartesian space, θ represents the joint variables (angles for revolute joints or displacements for prismatic joints), and f is the forward kinematic function. Computing the forward kinematics function, given a geometrical analysis of the robot manipulator at hand, is generally a straightforward process that can be solved analytically using techniques such as the Denavit-Hartenberg (D-H) convention [41].

Inverse kinematics, on the other hand, involves determining the joint configurations that will result in a desired end-effector pose. This is generally a more complex problem, as it may have

[1] As the field of robot learning evolves, however, we are seeing a more pervasive use of humanoid hands, that allow for a less restricted array of movements, more closely emulating human behaviour. Although most of the research works presented in this thesis are designed with a parallel-jaw gripper in mind, we will see how some final ones would allow for the use of humanoid hands, and could possibly even benefit from them.

multiple solutions, no solutions, or infinite solutions depending on the robot's configuration and the desired pose. As the forward kinematics function f is very often non-linear, and dependent on may trigonometric functions, inverting it is non trivial. Mathematically, inverse kinematics can be expressed as $\theta = f^{-1}(x)$, where f^{-1} represents the inverse kinematic function. There are two main possibilities to tackle the above: one is to undergo a mathematical and geometrical investigation of the robot configuration and therefore of f, to find a close form solution f^{-1}. [2] The second possibility, which is often the most used solutions by physical simulators, is to find a numerical solution through an optimisation algorithm. This approach would therefore sample and evaluate a large array of joint configurations, determine their "distance" from the desired end-effector pose by evaluating $f(\theta)$, and guide the search towards the best candidates.

In this thesis, we focus on *what a robot should do*: our final goal is generating a movement of the robot's end-effector that would successfully achieve a goal. We therefore have a series of assumptions in all of the work presented in this manuscript:

- We ignore the dynamics of the robot and focus on generating a trajectory of poses and/or velocities of the end-effector. We additionally assume the existence of both the forward and inverse kinematics functions f and f^{-1}, at least from a high-level perspective. We therefore assume the robot we use has a lower-level controller and kinematics functions and/or optimisation algorithms that, taking as input the current configuration and desired movement, can generate joint torques that will lead to accelerations and finally to the desired velocities and poses, once integrated.

- We focus only on the end-effector pose. While the robot could interact with the environment through any part of its chain of links, willingly or unwillingly, we ignore this problem, and assume the environment is not as cluttered, so that unwanted collisions between parts beyond the end-effector are rare. The field of collision avoidance proposes a large set of techniques to avoid such situations, which are, however, beyond the scope of this work.

We will instead focus on finding ways to map higher-dimensional sensory inputs, such as vision and natural language, to a desired trajectory of the end-effector.

[2] As the pose of a body in 3D space is represented by 6 variables, 3 for position and 3 for orientation, it is immediate to see that only a 6-DoF robot arm could allow for a unique, closed-form solution. Fewer DoFs would lead to no solutions to arbitrary end-effector poses, therefore, it is necessary to ignore one of the variables, while more DoFs can lead to infinite solutions, in this case allowing us to choose one of such solutions that can optimise another variable of interest, such as keeping all joints as far as possible from their configuration limits.

3.2 Seeing: Modelling Cameras as Robot Eyes

Vision is a fundamental sense for most living beings. Billions of years of evolution determined that the ability to sense and make sense of light rays (electromagnetic waves with a wavelength contained between 4×10^{-7} and 7×10^{-7} metres [157]) in our environments is a tremendously valuable ability for most species to survive. In primates, a large part of the brain is dedicated to processing vision [64]. Our artificial environments, and even everyday objects [148], have been strongly shaped around visual cues and our ability to interpret and react to them. It is therefore logical that robots that act in everyday environments need to be able to see and interpret what they see. In this section, we will focus on the former: *how can robots see*. Understanding what they see and how to react to that, just as it takes a large portion of many animals' brain, will later take a large portion of this thesis.

At the core of robotic vision systems are cameras, which serve as the "eyes" of the robot. In the current world, cameras are ubiquitous, to say the least. However, it is interesting to note that the fundamental structure of how a camera works has not changed substantially in the last few decades. The pinhole camera model provides a simple yet powerful mathematical framework for describing how three-dimensional scenes are projected onto a two-dimensional image plane. The pinhole camera model is based on the principle of how light interacts with objects and travels through space. Light rays[3] emanate from sources (such as the sun or the screen of your laptop) and bounce off surfaces in all directions. Eventually, rays reach the camera's aperture – in the idealised pinhole model, this is an infinitesimally small hole. Remarkably, each ray of light, after bouncing countless times from its original source to the aperture, transports a very tiny amount of information to the camera: its frequency is converted into three numbers, very often represented as *unsigned 8-bit integers* [101, 69], therefore representing three natural numbers between 0 and 255 included. Each number represents the intensity of, in many representations, the *red, green* and *black* elements of an arbitrary colour. As artificial light sources in electronic devices *emit* light, and therefore use an *additive* approach to generating colours, these numbers reflect the intensities of the red, green, and blue channels of each pixel used to generate the desired colour[4].

The way a point on an object in the three-dimensional world is seen by a pinhole camera is a function of three things: the position of the point in the world, the pose of the camera in the

[3]Understanding the fundamental physical nature of light is beyond the scope of this work. However, treating it as lights that travel in space in straight lines and bounce off objects is a useful model.

[4]The reason why we widely use the RGB formulation has roots both in the human optical perception system [26] and in the design of camera sensors and artificial light emitters like LEDs. [101]

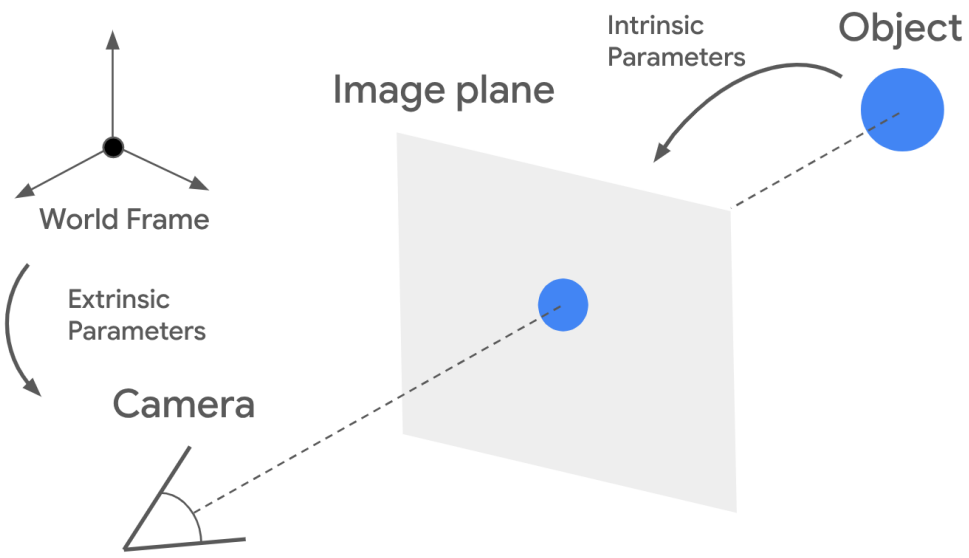


Figure 3.1: A minimalist illustrations of the pinhole camera model, highlighting the role of the extrinsic parameters, that define the pose of the camera in the world, and the intrinsic parameters, that define how an object will appear on the image plane.

world, and the internal structure of the camera. The first can simply be represented as a vector in homogeneous coordinates [86] $X = [x, y, z, 1]^T$. The extrinsic parameters, that compose the extrinsic matrix, define the camera's position and orientation in the world, effectively determining where the camera is looking. The extrinsic matrix is represented as a rotation matrix and a translation matrix, following the algebraic $SE(3)$ [86] conventions, $E = [R|t]$, expressed in the world frame. The internal characteristics of a camera, defined by the way it is designed and built, are represented by the intrinsic matrix. It is composed of 1) the focal lengths f_x, f_y, representing the distance between the sensor, i.e. the electronic device that can measure light and its waveform, and the aperture of the camera, the part where the light comes in 2) the principal points p_x, p_y, representing the coordinate, in the pixels frame, where the line passing through the aperture and perpendicular to the sensor, or image plane, intersects the latter. While additional parameters can be considered, in this thesis we consider a simplified version of the pinhole camera model, where these values entirely describe the structure of the camera. The intrinics matrix is composed as $K = \begin{pmatrix} f_x & 0 & p_x \\ 0 & f_y & p_y \\ 0 & 0 & 1 \end{pmatrix}$. To project a point X from world coordinates to a point x_p in 2D pixel coordinates (i.e. where it would appear on an image captured by the camera), it is sufficient to compute $p = KEX$, where $p = [p_x, p_y, p_z]$, and finally $x_p = [p_x/p_z, p_y/p_z]$ [86].

3.2.1 Beyond RGB: Measuring Depth with modern cameras

It is no coincidence that most animals possess more than a single eye. Albeit each eye is generally sufficient to perceive the surrounding environments visually, multiple eyes looking at the same targets allow us to estimate how far objects are through binocular vision [86], an ability that is very useful to interact with one's surroundings. While the field of binocular, or stereo, vision has led to a set of techniques to estimate depth from multiple cameras, engineers and scientists have found a different, less bio-inspired solution: depth cameras. Although many different technologies have been developed over the years [2, 4][5], whose physical workings are outside the scope of this work, the output of such devices is similar and of easy description: besides measuring the light coming from a point in space, these cameras can also estimate how far the object is from the image plane. This information is of great use for robotics, where knowing the location of objects in 3D is often of great aid to execute manipulation policies.

Therefore, cameras are sensors that translate the rays of light hitting their sensors into a matrix of numbers. How to make sense of these numbers is a crucial aspect of robotics, and in particular of robot learning.

3.2.2 Visual Servoing: Connecting Vision to Action

Visual servoing is a fundamental building block of the first half of this thesis. Visual servoing is a basic form of eye-hand coordination, connecting vision to actions: it refers to the use of visual feedback to control robot motion, where the error signal driving the controller is computed directly from image features. This paradigm is central to several methods presented in this thesis, particularly the alignment phases in Chapters 4-6. Image-Based Visual Servoing (IBVS) [32] computes control commands directly in image space. Given a set of image features s (such as point coordinates, line parameters, or image moments) and a desired feature configuration s^*, IBVS minimises the error $e = s - s^*$ through a velocity controller: $v = -\lambda \ \mathtt{L}^+ e$ where $\mathtt{L}^+$ is the pseudo-inverse of the image Jacobian (or interaction matrix) $\mathtt{L}$, which relates changes in image features to camera/end-effector velocities, and λ is a gain parameter. Position-Based Visual Servoing (PBVS) [32] instead reconstructs the 3D pose of objects from visual observations and controls the robot in Cartesian space. While potentially more intuitive, PBVS requires accurate camera calibration and 3D reconstruction. The methods in this thesis primarily employ variants

[5]It is interesting to note how the Kinect, developed by Microsoft, was first designed as a consumer product aided to the videogames industry. The core idea was to allow users to interact with software by moving their body in space. The product saw little success[3], but unintentionally became one of the favourite tools of roboticists.

of IBVS using learned feature representations. Rather than hand-designed features like corners or edges, we use features extracted from deep neural networks, described in later sections, which provide robust correspondences across object instances and viewpoints. The "alignment" phases described in Chapters 4-6 can be understood as learned visual servoing [175] controllers that minimise the discrepancy between current and goal visual features. It however happens in two different ways: Chapters 4 and 5 use neural networks to learn visual servoing end-to-end, learning how to perform visual servoing given raw image inputs. In Chapter 6, we use a pretrained model to extract visual keypoints, that allow us to analytically compute the desired robot movement that performs visual servoing error reduction as above. We will again use keypoints extensively in Chapters 7 and 8, but used to design methods that go beyond visual servoing.

Keypoints provide a sparse, structured representation of visual observations that has proven particularly effective for robot learning. A keypoint is typically defined as a 2D or 3D spatial location corresponding to a semantically or geometrically meaningful part of an image (e.g., the corner of an object, the tip of a tool, a distinctive texture patch).

Deep Spatial Autoencoders [65] learn to extract keypoints in a self-supervised manner by training an encoder-decoder architecture where the bottleneck layer is constrained to represent (x, y) coordinates. The decoder must reconstruct the input image from only these coordinates, forcing the encoder to select maximally informative spatial locations. This approach is used in Chapter 4. Correspondence-based keypoints (used in Chapters 6-8) leverage vision foundation models to find semantically meaningful correspondences between images. Given a reference image and a target image, dense features are extracted from both, and keypoints are defined as locations with matching feature descriptors. The use of large pretrained models to extract correspondences allow us to use the semantic world understanding distilled into the networks, and find correspondences between semantically similar, but possibly very visually different parts of objects (e.g. the tip of two different shoes). A deeper description of such methods will be provided in the next sections and chapters.

3.3 Communicating: Representing Language as Bits

While industrial robots are programmed to execute one or a few tasks, the generalist robots we envision should possess the ability to execute a large series of tasks in many environments. Just as we humans, they therefore need to be able to understand commands and communicate.

[6] In addition to the sole act of verbally communicating, humans can also store information via writing, which led to the creation of a vast pool of knowledge from which humans and machines can learn. To both understand humans and the vast knowledge that is stored in language, robots can be equipped with techniques from the field of natural language processing (NLP), that can be seen as a branch of artificial intelligence. A comprehensive introduction to the field of NLP is beyond the scope of this work, but we will here describe how language is represented in digital devices, and will later mention the main works that show how to incorporate language understanding in robots.

Just as numbers are represented as bits in computer memories, so are characters that compose words and text. Many representations have been proposed over the decades, from ASCII [1] to the more recent Unicode [5], each being a convention for a mapping between characters and an arbitrary sequence of bytes.

It is common to preprocess text via tokenization, instead of representing it at a character level. A token is an integer that represents a sequence of characters that can be a word or a part of a word. The tokens used to represent language, being a set of discrete elements stored as numbers, are therefore the vocabulary that can be used by an algorithm or model. Although single characters might be used as tokens, as was often the norm [187], since they indeed represent the atomic building blocks of language, the recent literature has demonstrated that (sub)word-level tokens often lead to better performance [80].

Once words and sentences are tokenised and converted into bits, and therefore numbers, ideally any technique of data analysis, processing, and machine learning can be used to learn arbitrary mappings from these numerical inputs to any numerical output of interest, such as determining the sentiment of a sentence, what word is likely to come next, or what actions to execute given a language command, as we will see in later chapters.

3.4 Learning: Finding Sense from Data

We introduced the two ends of a robotic agent: we started from the output, the actions. While the output of many computer programs is a novel configuration of bits stored into a memory device, the output of a robotic task should be a novel configuration the environment around it. The robot should manipulate not bits, but in a certain sense, atoms. We then described

[6] While we argued that vision occupies a large part of the brain of primates and is possessed by a large variety of animals, who strongly rely on it for survival, the same cannot be said for language. Although many animals exhibit peculiar ways of communicating [75], the bandwidth of these modalities is nowhere near human language.

what inputs we can expect a robot to receive, and in particular the ones we will focus on in this thesis: vision, fundamental to understand the environment around the robot, and language, used in this case only to guide the robot execution. What is missing now is the link between these modalities, something that can process these inputs once converted into numbers (we saw that cameras transform rays of lights into matrices of integers, and language is transformed into a sequence of tokens) to meaningful actions.

As we discussed in previous chapters, the field of AI saw a dramatic paradigm shift with the evolution of machine learning. We also saw how robotics was often influenced by the breakthroughs in AI. Initially, roboticists focused on designing explicit algorithms and heuristics to solve specific tasks. This approach, while effective for well-defined problems, often struggled with the complexity and variability of real-world scenarios. Manually finding techniques that can map such highly dimensional inputs, like images, to sensible actions proved to be extremely challenging beyond trivial tasks in controlled environments. As machine learning techniques matured, the field began to adopt a hybrid approach, combining traditional algorithmic solutions with learnt components. This phase was characterised by extensive feature engineering, where domain experts would design relevant features for learning algorithms to process. Eventually, the progressive growth of datasets, computing power, and understanding of techniques that led to efficient machine learning models led to an increase in reliance on end-to-end solutions, where a model could learn to extract and process such features autonomously from raw inputs. In this section, we will walk through this evolution, introduce the fundamental building blocks that we use in this thesis, and explore how the field of robot learning has evolved over the last couple of decades.

3.4.1 Neural Networks: A Brief History, and What Made Them Work

A neural network is fundamentally a parametric function approximator. It computes a numerical output given a numerical input, and can shape the function mapping the latter to the former by modifying its parameters, or weights. Given a dataset of desired input and output pairs, the weights of neural networks can be modified via an optimisation procedure to mimic the desired mapping demonstrated in the data. The concept of artificial neural networks dates back to the 1940s, inspired by our understanding of biological neurons. However, it was not until 1957 that Frank Rosenblatt introduced the perceptron [171], a simple classifier that laid the foundation for modern neural networks, based on the theory presented in [140]. It is remarkable that the

perceptron was effectively a physical neural network, with weights encoded as potentiometers and modified by electric motors. Notably, for a considerable time only the last layer of perceptrons was trained, strongly reducing its potential [171, 176]. More general algorithms to train multi-layer perceptron started to appear from 1970 in [127], where an early version of fundamental algorithms was presented: backpropagation, further investigated in [174].

The structure of the perceptron is still widely used in the design of current neural networks. A feed-forward neural network, often referred to as a multi-layer perceptron (MLP), takes as input a one-dimensional vector $x \in \mathbb{R}^{1,N}$ where N is the number of input features. The network can then compute an arbitrary number of features from this input, that will form the units of its first hidden layer. This intermediate hidden output is computed as $h_i = f(W_i x + b_i)$, where f is a non-linear activation function, W_i is a $M \times N$ matrix, where M is the desired number of hidden outputs, and b_i is a $1 \times M$ bias vector. This process can be repeated an arbitrary number of times, with x becoming h_i. Finally, the last layer will compute the desired shape of the output. A feed-forward neural network is therefore a series of matrix multiplications, interleaved by the use of non-linear activation functions. Remarkably, the Universal Approximation Theorem [95] states that a feed-forward network with at least one hidden layer can approximate arbitrary functions. Therefore, these models seem suitable for learning arbitrary input and output mappings. Albeit the theorem states the existence of such a network as a combination of weights, it does not provide a way to find such weights. To gradually modify them in order to shape the represented function towards the desired mapping, we need to rely on multi-variate derivative computations and optimisation techniques, such as backpropagation and gradient descent.

Backpropagation is "*essentially an efficient way of implementing Leibniz's chain rule* [176]": in order to understand how to change each parameter, or weight, of a neural network, we need to compute the gradient of the objective we want to optimise with respect to each weight θ_i. The objective is often a *loss function* we want to minimise, $L(\hat{y}_\theta, y)$, where $\hat{y}_\theta$ is the output of the network given its weights θ and input, and y is the desired output. This optimisation strategy, in which we compare the model's output to a ground truth value we want to obtain given an input, is called *supervised learning*, and forms the backbone of modern machine learning. Backpropagation computes "backwards" the gradient of the loss function with respect to the weights of the last layer, and then further computes the gradient of the inputs of these layers with respect to the weights of the layers before, and so on. Through this chain of derivatives, we can compute all desired gradients $\delta L / \delta \theta_i$, where θ_i is an arbitrary weight of the network. We then perform gradient descent: we modify slightly all weights in the direction that minimises

L, therefore taking the opposite direction of $\delta L/\delta\theta_i$.

Although we now that the training of neural networks through backpropagation and gradient descent led to a series of unprecedented breakthroughs in many fields [77], the history of training such models has not been a linear path to success. For many years after the original development of the perceptron, supervised training of neural networks through backpropagation was seen with scepticism, and other techniques were preferred. A series of breakthroughs [120, 116, 88], mainly in the field of computer vision, demonstrated how such techniques could not only scale to more complex, highly dimensional inputs, but also to unprecedented size and depth of these neural networks, which fundamentally improved their performance [76].

What made neural network training finally work, leading to the emergence of deep learning? [77] While a comprehensive history of technical, theoretical and empirical findings is beyond the scope of this work, we here briefly describe some fundamental aspects that affected their evolution.

- **Initialisation strategies**: A fundamental issue of training deep networks has often been the collapse in the output distribution of several deeper units. As investigated in [76], the use of better designed activation functions can strongly improve training stability, keeping the outputs all of units into desirable ranges. This paper proposed one of the most used initialisation strategies, the Xavier Initialisation, that shed light on the issue of correctly initialising networks' weights.

- **Activation functions**: As also discussed in [76], activation functions that quickly saturate, like the sigmoid function, can lead to the collapse of the output distribution of many units. In deep networks, the use of more linear units, like the Rectified Linear Unit (proposed in 1969, but started to be widely adopted almost half a century later) [71], led to better training convergence. Over the recent years, many variations have been proposed [115, 90].

- **Normalisation techniques**: A wide range of works investigated, as we have seen before, how the main reason for the failing of networks training can be traced back to a collapse in the output distribution of units. Besides the techniques proposed above, a series of normalisation techniques were also designed over the years. These techniques modify the output of hidden layers of neural networks by shifting and manipulating their mean and variance. One seminal work in this field is [103], that proposed a technique to keep the output distribution of units stable and into a desired range, Batch Normalisation.

Different techniques have been proposed over the years, with Layer Normalisation [15] being one of the most used today.

- **Residual Connections**: A significant challenge in training very deep networks is the vanishing gradient problem, where gradients become increasingly smaller as they are back-propagated through many layers, hindering learning in earlier layers. This issue makes it difficult to train networks with a large number of layers effectively. Residual connections, introduced in [89], provide a solution by adding "identity shortcuts" that directly connect the input of a layer to its output. Instead of learning a direct mapping from input to output, each layer learns a residual mapping, effectively learning how the output differs from the input. Formally, if we denote the desired mapping of a layer as $\mathcal{H}(x)$, a network with residual connections learns a function $\mathcal{F}(x) = \mathcal{H}(x) - x$, and the stacked layers learn $\mathcal{F}(x) + x$. This seemingly simple modification has profound implications. By allowing gradients to flow more directly through these shortcut connections, residual connections alleviate the vanishing gradient problem, enabling the training of significantly deeper networks. This breakthrough allowed for the creation of networks with hundreds or even thousands of layers, leading to substantial performance improvements in various tasks, particularly in computer vision. The seminal ResNet architecture [89] demonstrated the power of residual connections, achieving state-of-the-art results and fundamentally changing the landscape of deep learning by making it possible to effectively train extremely deep neural networks.

- **Optimisers**: Backpropagation computes the chain of derivatives that informs what direction should the weight locally move towards to minimise the loss function. The way the weights are then effectively modified depends on the optimiser we employ. The simplest implementation is Stochastic Gradient Descent [78], where the loss function is computed on a random sample of the entire dataset, and weights are updated towards the direction of loss minimisation, by multiplying each weight derivative for a learning rate δ. However, these may suffer from local minima, as they are generally unable to espace saddle points where gradients are close to zero. More modern optimisers, like Adam [114], adapt the learning rate of each weight based on their recent history of change, avoiding drastic changes of few parameters, and including a momentum parameter that exponentially integrates the variation of each weight to keep their movement even when facing saddle points where the gradient are near zero. These advancements in optimisers have been instrumental to speed up training of deep networks, and ensure smoother convergence.

- **Datasets and Compute**: While these are not inherent new scientific and engineering findings in the design space of neural networks, the exponential increase in the size of datasets and in the computational power of hardware accelerators has been one of the strongest pushes towards the unprecedented capabilities modern neural networks display. Larger datasets unlock the use of larger networks with billions of parameters without the fear of overfitting, and powerful hardware accelerators allow to train and experiment orders of magnitude faster than just a decade ago.

  Notably, recent state of the art models in computer vision and natural language processing are nowadays trained with batch sizes, the number of individual training samples used to compute and average the gradients, orders of magnitude larger than what was the standard few years ago. [116, 29] This is only possible when the datasets are enormous, and the hardware accelerators are able to store tens or hundreds of GBs of data for each feed-forward pass.

3.4.2 Architectures Used in this Work

The evolution of neural network design encompasses decades of research. As we briefly described, a plethora of improvements, discovered over the years, unlocked the power of neural networks as powerful and versatile function approximators. While a comprehensive description of neural networks architectures is beyond the scope of this work, we will here give an intuitive introduction to the architectures we use in this work.

All the research described in this thesis is based on one of the following architectures: feed-forward neural networks [171], convolutional neural networks [120], and transformers [193]. To better describe their properties and differences, we will show how they compute outputs given the same input. We will then use this example to examine and list more in detail how these networks operate as the inputs change in size, and how their dimension and outputs scales accordingly. We will assume the input is a vector i of dimension $1 \times N$.

Fully-Connected Networks (FCNs), often called Multi-Layer Perceptrons for historical reasons [171]. They fundamental computational operation is the linear matrix multiplication. Each layer of a FCNs is represented by a matrix of weights $W \subset \mathbb{R}^{M,N}$, where N is the size of the input as described above, and M is the dimension of the output, arbitrarily chosen. A layer of an FCNs computes its output through the following stages: it first computes the output vector o of dimension $1 \times M$ as $o = Wi$. It then adds to o a bias vector b of dimension $1 \times o$, and finally applied a non-linear activation function f. Overall, the output is $h = f(Wi + b)$.

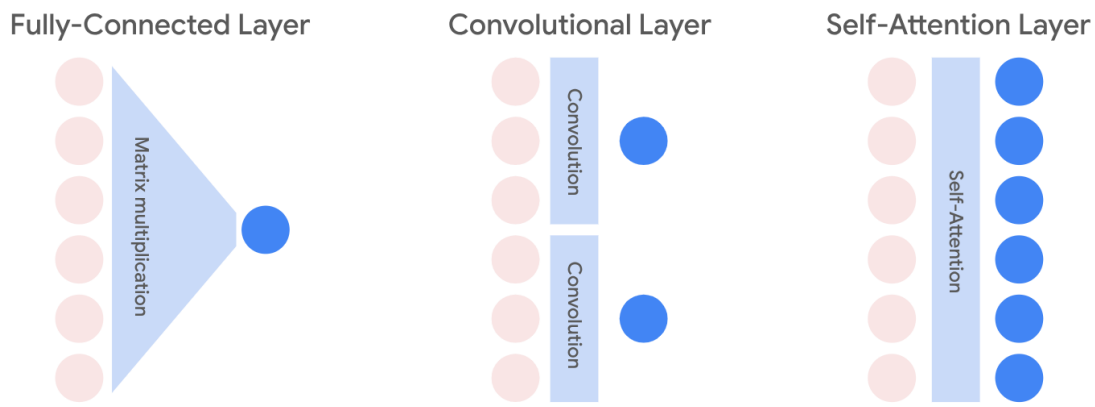


Figure 3.2: A minimalist illustrations of the core layers that define the three main network architectures used in this work. The illustration serves to elucidate the structural relations between the input dimensionality and the output dimensionality of these layers. Each layer is composed of the inputs on the left, the mapping operation in the center, and the outputs on the right.

We can list a series of observations regarding this architecture: **1)** they expect an input of fixed dimension **2)** the size of each layer scales linearly with the size of the input and of the desired dimension of the output **3)** the weights W of each layer are applied to the inputs in the same, fixed order: any shuffling or shifting of the input features would lead to a completely different output. FCNs are therefore not equivariant or invariant to any manipulation or reordering of the inputs.

Intuitively, FCNs are an effective tool to learn functions that map 1-dimensional vectors to other 1-dimensional vectors, like discrete categories probabilities when dealing with classification tasks, or scalar outputs for regression tasks. They are therefore largely used in robotics, by themselves or as parts of a more complex architecture, to map intermediate representations to, for example, joints or cartesian velocities and positions.

Convolutional Neural Networks (CNNs) were designed taking inspiration from the visual system of many animals, such as cats [100, 99]. These networks are composed of convolutional filters, whose design was perfected over the decades [72, 120, 116], and whose popularity exploded after they demonstrated unprecedented performance on computer vision benchmarks [116, 37]. They architecture and computational graph revolve around the convolutional filter layers, a patch of a few weights that perform local multiplications over the input through convolution[7]. Each layer of the convolutional network is composed of a series of filters, or kernels.

[7]It is interesting to note that, technically, the operation performed by these networks is the cross-correlation, not the convolution, where generally the filter would need to be mirrored first. However, it is standard to call the operation convolution.

Given our example 1-dimensional input $i \subset \mathbb{R}^N$, the kernels would have shape $1 \times L$, also being 1-dimensional, with $L < N$. These filters are applied to the input by applying a convolution between the two [120, 116]. The output is a 1-dimensional vector o of shape $1 \times M$, where $M = N - L + 1$, assuming zero padding and stride 1[8]. We represent the convolution operator between input i and kernel k as $o = i * k$. As for the FCNs, each kernel adds a scalar bias b after the operation, and finally applies a non-linear activation function f, $h = f(i * k + b)$. When many kernels are present in a layer, the outputs are concatenated along the first dimension: K kernels applied to i would lead to output o_k of shape $K \times M$.

There are some substantial differences with FCNs, that we can highlight by listing the same properties we listed before: **1)** convolutional layers can be applied to inputs of varying size, as the convolutional operation involves applying a fixed filter to all the features of the input **2)** the size of the layers is independent to the length of the inputs, for the same reason mentioned before, while the computational complexity scales linearly due to the convolution operation **3)** remarkably, as the kernel filters are applied through a convolution operation, convolutional layers are shift-equivariant. If the input features are shifted by ΔP positions, the output will be shifted by the same amount. These makes convolutional features especially well suited to find local patterns in data, such as time-series in the 1-dimensional case, or images in the 2-dimensional case. Many CNNs, however, use fully-connected layers at the end of the computational graph, partially losing the shift-invariant properties.

Thanks to the aforementioned properties, CNNs led to major breakthroughs in the field of computer vision, and are largely used in many applications involving computer vision, as we will see in the next chapters.

Transformers were introduced in [193] as an architecture designed to process sequential data, particularly for natural language processing tasks, although their use has since expanded to many other domains. Unlike FCNs and CNNs, transformers compute representations of their inputs through an attention mechanism that weighs the importance of different parts of the input sequence relative to each other. Given our example input vector i of dimension $1 \times N$, a transformer first projects it into three different vectors through learnable linear projections: a query $q = W_q i$, a key $k = W_k i$, and a value $v = W_v i$, where $W_{\{q,k,v\}}$ are matrices of learnable weights. The fundamental operation of transformers is self-attention: given an input set of M vectors shaped like i, we first stack all the outputs of the previous operations row-wise into the matrices Q, K, V of shape $M \times N$. Self-attention then computes the attention scores

[8]Padding refers to adding 0s "surrounding" the input vector to let the output of the convolution have the same shape as the input. Stride refers to the step size of the shifting of the filter during the convolution operation[116, 88].

between each element of the input and all other elements as $A = \text{softmax}(QK^T/\sqrt{d})$, where d is a scaling factor [193]. These scores are then used to compute a weighted sum of the values: $O = AV$. This process allows each position to attend to all other positions, capturing global dependencies in the input sequence. As in the previous architectures, a non-linear activation function f is applied after adding a bias term b: $h_i = f(o_i + b)$, where o_i represents each row of O. We can analyse this architecture through the same lens we used before: 1) transformers can handle variable-length inputs, as the attention operation can be computed between sequences of arbitrary length 2) the size of each layer is independent of the length of the input sequence, but the computational complexity scales quadratically with the input length, as attention scores need to be computed between each pair of input elements 3) transformers are permutation equivariant[9]: shuffling the order of the input sequence will result in the same output being similarly shuffled, as the attention mechanism treats the input as an unordered set of elements. A notable difference from previous architectures is that transformers can capture long-range dependencies in the input, as each element can directly attend to any other element, regardless of their distance in the sequence. The ability of transformers to process sequences of arbitrary length while capturing global dependencies has led to their widespread adoption in natural language processing, computer vision, and robotics. In particular, their capacity to jointly process multiple modalities, such as vision and language, has made them particularly appealing for robotic applications where multiple sensory inputs need to be processed together to determine appropriate actions.

The tools we described are used as very plastic and expressive function approximators in order to let robot learn mappings between highly dimensional sensorial inputs and actions to be performed in the world. By letting efficient optimisation algorithms shape the weights of these networks by ingesting datasets of input-output pairs that represent the desired behaviour, roboticists can avoid designing complex hand-engineered algorithms to guide the movement of robots. Instead, one or more neural networks can learn to emulate the desired behaviour. We will now dive deeper into the recent decades of research that investigated techniques to improve the performance of these learning models.

[9]While the core attention mechanism itself is permutation equivariant, transformers typically include positional encodings [193, 161, 29, 73] that are added to the input embeddings to make the model sensitive to sequence order. Positional encodings break permutation equivariance, allowing transformers to learn and leverage sequential patterns in the data, such as sequences of images or actions.

3.5 Imitating: Learning Behaviour from Data

Imitation learning (IL), also known as learning from demonstration (LfD), represents a paradigm shift in how we approach the programming of robotic systems. Rather than explicitly coding every possible action and reaction, imitation learning enables robots to acquire skills by learning to emulate behaviours recorded by humans. Datasets are often stored by letting a human expert take physical control of a robot, and recording both the inputs from different sensors, such as cameras, force-torque sensors, and more, and the movement of the robot as guided by the human. A model trained on this data should then be able to emulate the behaviour implicitly demonstrated by the user. This leads to a fundamental different approach: instead of focusing on designing a hand-written algorithm that can perform the desired tasks, users now need to gather examples of the desired behaviour covering as many situations as possible that the robot may face when deployed. This strategy has been studied for many decades, as data collection advice was presented in [160] in 1988. This approach bridges the gap between human intuition and machine execution, allowing a more natural and efficient transfer of complex tasks to robotic systems.

3.5.1 Early Successes

One of the first breakthroughs of imitation learning was presented in [160], where a neural network was trained to drive a car from RGB images of the road and input from the laser range finder. This work presented many fundamental ideas that are still at the core of imitation learning: a neural network is trained on expert data, demonstrating how to drive and, therefore, behave, given the current sensory inputs that observe the environment. Notably, [160] already highlighted how the advantage of representing the entire policy in a neural network optimised via imitation learning is the autonomous emergence of features that can interpret both RGB images and laser range finder readings. Manually designing features that could extract actionable signal from these highly dimensional data streams would be challenging. Instead, through imitation learning, the user only needs to focus on gathering data that is varied enough to let the network learn to drive and recover from possible mistakes. Collecting data that demonstrate recovery from error, or more generally can tackle a shift in the input distribution, has been an active area of research for decades, as errors in action prediction, when an agent is deployed, can accumulate, leading to strong deviations from the optimal states distribution [172, 119, 45].

In the field of robot manipulation, several works emerged in the 1980s to introduce techniques that allowed non-experts to guide a robot, as an algorithm analysed the movement to

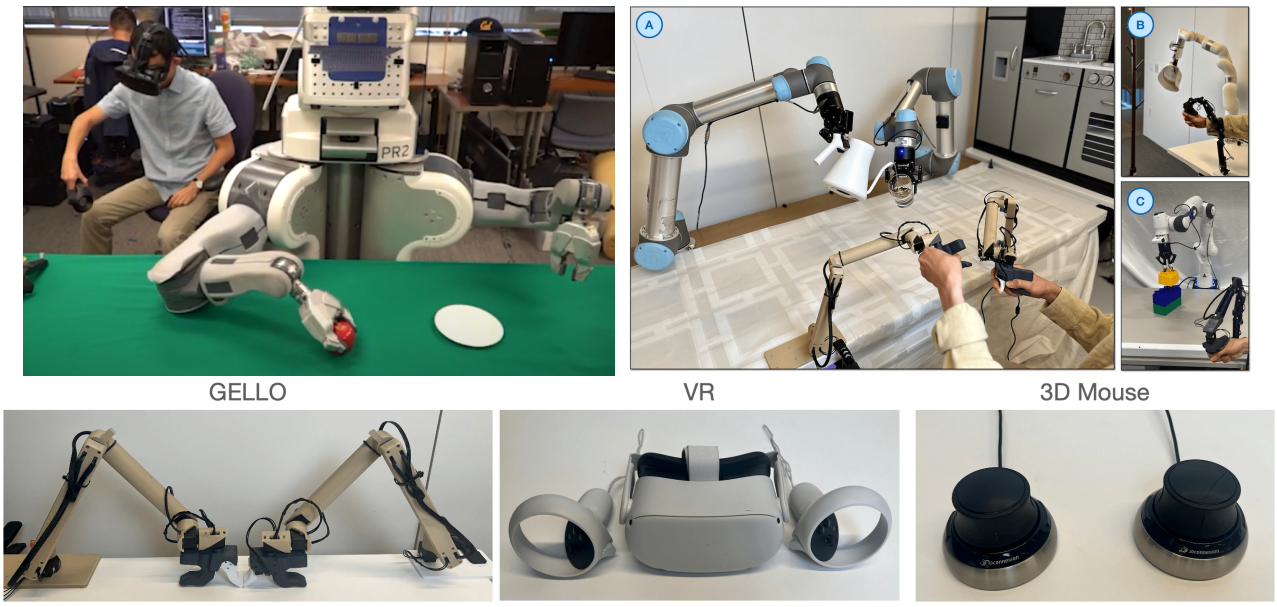


Figure 3.3: Top: Examples of users teleoperating robots. On the left (from [219]), a user teleoperates a robot through a Virtual Reality (VR) headset and controllers. On the right (from [210]), a user teleoperates a robot using kinematically similar arms, effectively puppeteering the robot that will mirror either the desired gripper poses, or each joint angle. Bottom (from [210]): a series of common controllers to teleoperate robots.

then replicate the behaviour autonomously, as in [122, 27]. However, these methods strongly relied on heuristics to transform a trajectory into low-level instructions that concatenated primitive actions, mainly pick and place movements. During the 1990s, the field of control and robotics saw a larger adoption of machine learning models and neural networks [84, 19]. The use of generating an entire trajectory from sensory input was not common, but parts of the robot control stack started to see the adoption of learnt modules, such as the proposal of inverse kinematics solutions [19], or learning torque controllers for multi-link arms to follow predefined joints trajectories [125].

3.5.2 From Pixels to Actions: End-to-End Neural Computing

In the following years, the increasing performance of neural networks in processing and understanding high-dimensional data such as vision [120, 116, 160] inspired researchers in robotics to design architectures that could accept these modalities as input. A series of remarkable works sparkled a much greater interest in the use of large neural models in robot control from vision: [144] demonstrated that a neural network could learn to master many Atari games receiving as input pixels alone, and [123] proposed an end-to-end architecture that could receive camera observations as input and output torque controls that would guide a robot arm to solve a variety of

manipulation tasks. Similarly, also the field of autonomous veichles, that had already adopted end-to-end neural solutions to represent policies, scaled up their models so that they could interpret larger, more complex image observations, closely mimicking what a human driver would encounter [21]. The field of robotics started incorporating large networks more and more as end-to-end solutions that could make sense of high-dimensional data and map those inputs to actions. As we described earlier, this led to a shift from designing policies to designing ways to collect data. The main approaches consisted mainly of "robot farms", that is, a large number of unsupervised robots trying actions and collecting data [112, 130, 108, 42], or collecting large quantities of data by teleoperating robots [104, 134] (examples of data collection can be seen in Figure 3.3).

3.5.3 Multi-Task, Multi-Modal Robots

As the robotics literature saw several examples of success cases in robots learning non-trivial tasks purely from vision [123, 21], the goal became to design even more general architectures that could solve a wide range of tasks, receiving commands in the form of goal images or language. The biggest and most constant lesson of machine learning and deep learning in the previous decade was one: large, plastic neural networks were able to autonomously learn complex mappings between very different input and output manifolds [208]. Furthermore, novel designs from any field of AI, such as NLP, speech processing, or computer vision, were easily transferable to the robotics field [193, 208, 88]. Therefore, the robotics literature started using processing techniques and network designs, not only from the computer vision field but also from the NLP field, to allow robots to learn to map images and language to actions. By gathering datasets of *(images, language)* $\rightarrow$ *(actions)* pairs, the field started training large multi-modal networks to guide robots during the last few years. These efforts led to a set of works that demonstrated the abilities of robots to be controlled through language commands [131, 133, 104] and goal images and videos of desired behaviour [104, 132]. This recipe has been at the core of a large part of the robot learning literature, in an effort to replicate the breakthroughs that were seen in other fields of AI and that therefore confirmed that large neural models were able to learn these complex mappings from supervised learning, leading to large scale projects [28], some of which involved an unprecedented data collaboration between tens of institutions around the world, including ours [40].



Figure 3.4: Robots that operate in human spaces will encounter complex environments, and will need to understand nuanced instructions. Image taken from deepmind.google.

3.5.4 Transferring Knowledge Across Tasks: Building on Foundation Models

Despite the various successes, the field of robot learning has long been constrained by the slow and costly process of collecting robot data. Unlike computer vision and NLP, where vast datasets are readily available, robotic data collection, as we saw, is often time-consuming and requires substantial hardware investments and efforts [104, 108]. This observation inspired research efforts to find ways to incorporate the wealth of data and pre-trained models from these adjacent fields into robotics. This shift has led to an increased interest in the use of foundation models. large-scale models pre-trained on diverse, Web-scale datasets, to enhance robotic capabilities and reduce the need for robotics-centered data [22]. One of the primary areas where foundation models could be transferred to robotics is visual perception. Large-scale vision models, pre-trained on large, diverse datasets, have shown remarkable capabilities in understanding and interpreting complex visual scenes. Most importantly, they have been shown to be able to extract useful and general representations [162, 30, 152] from images that can strongly speed up the training of neural models that map these representations to, for example, actionable outputs. For example, [182] design a two-stream architecture, combining a learnt "action" branch with a frozen, visual, semantic branch, based on a pre-trained CLIP model

[162]. This results in a method able to quickly learn new policies, and that could generalise very effectively to novel visual scenarios, by leveraging the representations extracted by the vision model. Other papers used the abilities of Large Language Models (LLMs) [29, 61, 188] to equip robots with their reasoning ability, unlocking the possibility to communicate with robots using natural, unstructured language commands, and having the robot reason about such commands, breaking them down into lower-level sub-goals that could be more easily grounded into actions [60, 48]. Finally, a series of recent works used general, multi-modal Vision Language Models (VLMs), finetuning them on robotics data, to obtain models that could effortlessly interpret images and language and output actions directly, without the need to connect different neural models or design ways to ground into actions the output of other blocks [56, 52].

In our thesis, we present a series of works that are focused on tackling some of the aforementioned challenges: designing techniques that reduce the need for human effort in collecting robot data, mostly by leveraging data autonomously collected by the robot, and designing methods to more fully leverage the abilities and properties of recent vision and language foundation models.

This chapter served as a general, high-level introduction to the technical concepts we will use in this thesis, and to walk the reader through the main concepts, papers and ideas of the robot learning literature over the last decades. The next chapters will also include additional Related Work sections that will explore more vertically some of the most important and influential works of the literature related to the specific topics and techniques investigated in each of these chapters.

Chapter 4

Learning Multi-Stage Tasks with One Demonstration via Self-Replay

4.1 Introduction

One of the principal, long-term goals of robot learning, is to enable a robot to learn a new skill from human demonstrations in a simple and efficient way, without significant effort from the human, and without them requiring expert knowledge of the underlying algorithm. Imitation learning techniques have been widely adopted in recent years, but these methods often require a considerable human effort, as we have seen in previous Chapters. In Sec. 4.2, we describe in more detail the shortcomings of techniques like behavioural cloning, meta learning, and reinforcement learning, that have hindered their widespread adoptions outside of laboratory settings. Several manipulation tasks involve multiple stages, e.g. grasping and then using a tool (Fig.4.1), or re-arranging a set of objects. But generally, the more stages a task involves, the more difficult it is to learn [82, 168], further exposing the limitations of the aforementioned techniques.

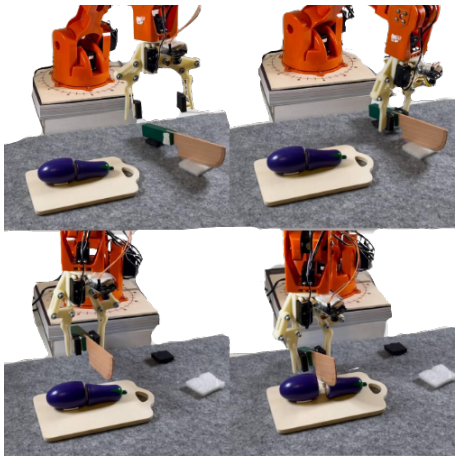


Figure 4.1: An example of a two stage task: picking up a knife, and then using it to cut a toy vegetable.

In this work, we introduce a method that allows a human operator to **teach a robot to solve a new, multi-stage manipulation task, with a single demonstration**, without the need for any prior data or knowledge about the object. Our method is capable of solving a series

of everyday-like tasks, and can be effectively used by a non-expert, given its simplicity and need for minimal human effort. The main assumption is that most everyday manipulation tasks can be divided into two phases: a coarse, object reaching phase, and a fine and precise, object interaction phase (Fig. 4.1, 4.2). As such, a multi-stage task can be decomposed into several stages, each of which alternates between these two phases. This decomposition forms the basis of our method, which allows the robot to collect all the experience it requires autonomously, without requiring further human supervision, such as repeated environment resetting. We call our method **Self**-Supervised Learning with Actions **Replay**, or **Self-Replay**. Self-Replay is inspired by the Coarse-to-Fine Imitation Learning method [110], which also models tasks with coarse and fine trajectories, but which can only solve single-stage tasks. Here, we develop a more general method that can learn a wide variety of multi-stage tasks, whilst still only requiring a single human demonstration.

Self-Replay is simple but effective. We evaluated our method on a set of real-world, multi-stage tasks, inspired by everyday primitive skills such as object grasping, object placing, object pushing, shape insertion, and food cutting. Our method is able to solve multi-stage combinations of these skills, whereas in the literature such tasks generally require at least tens of demonstrations, or repeated manual environment resets, or engineered laboratory setups [218, 217, 123, 67, 165]. We also provide a series of ablation studies and comparisons with existing techniques in the literature, to further understand the contribution of each component of our method. Our method only requires an uncalibrated wrist-mounted camera. While the training should take place in an uncluttered environment, we demonstrate how our method can tackle visual distractors at test time.

Our contributions are the following: (1) We introduce Self-Replay, a novel self-supervised learning method to solve everyday, multi-stage manipulation tasks from a single human demonstration, with no additional human effort, prior knowledge, or engineering needed. (2) As well as extending it to multi-stage tasks, we extend the data collection method proposed in [110] with a more efficient, active variant, that we call Active Self-Replay. (3) We designed a novel test-time pipeline that allows the robot to tackle multi-stage tasks, whilst also recovering from external disturbances and errors, and being robust to visual distractors. (4) We benchmark our method on a varied set of real-world, everyday-like manipulation tasks, and compare it against other methods from the recent robot learning literature.

4.2 Related Work

The recent imitation learning [74, 153, 102, 173] literature has proposed a series of techniques to teach a robot a novel skill, starting from demonstrations, in an efficient way. However, most techniques often require a considerable amount of work, prior knowledge, or supervision from a human operator. Behavioural cloning methods [45, 218, 132, 217, 67, 123, 82, 65, 119] learn a policy network directly from human demonstrations. While providing demonstrations is conceptually easy, simple tasks may require tens or even hundreds of them, making the process laborious. When dealing with multi-stage tasks, the amount of demonstrations often increases considerably [82, 132]. Meta learning methods [54, 107, 213, 66, 212] require one or few demonstrations to learn to solve a new task. However, they need large amounts of data collected before hand on similar tasks, another laborious process, and the amount of previously collected data also increases when tackling multi-stage tasks [212]. Plus, they can often only adapt to tasks that are very similar to what was observed during meta-training [54, 107, 213, 66]. Reinforcement learning methods [168] can use demonstrations to bootstrap the agent's policy [194, 146] or shape its reward function [92, 70]. Even if they require a few demonstration, the autonomous, reinforcement learning phase needs constant human supervision, as the environment need to be reset after each episode. Additionally, a reward function needs to be shaped and provided at each time-step to the agent, which may require additional engineering of the setup [168]. To solve multi-stage tasks, the recent literature has also proposed several methods that combine ideas from different fields: auto-regressive sequence modelling [158], neural task programming [211], exploiting spatial symmetries [216], hierarchical decomposition [82], explicit task planning [96], etc., but each technique requires considerable human effort in terms of either data collection or engineering. Our method builds on a long tradition of visual servoing for robot manipulation. Classical image-based visual servoing (IBVS) [32] uses hand-designed image features and analytical Jacobians to close the control loop. However, these methods require careful feature engineering and calibration. Learning-based visual servoing has emerged as a promising alternative. [175] trained deep networks to predict visual servoing commands for grasping, but required extensive labelled data. [65] demonstrated end-to-end learning of visuomotor policies, effectively learning an implicit visual servoing controller, but required months of robot time for data collection. The Coarse-to-Fine framework [110], which our method extends, introduced the key insight that separating the servoing problem (reaching the bottleneck pose) from the manipulation problem (interacting with the object) dramatically improves data efficiency. FlowControl [13] uses learned optical flow for visual alignment, achieving impressive

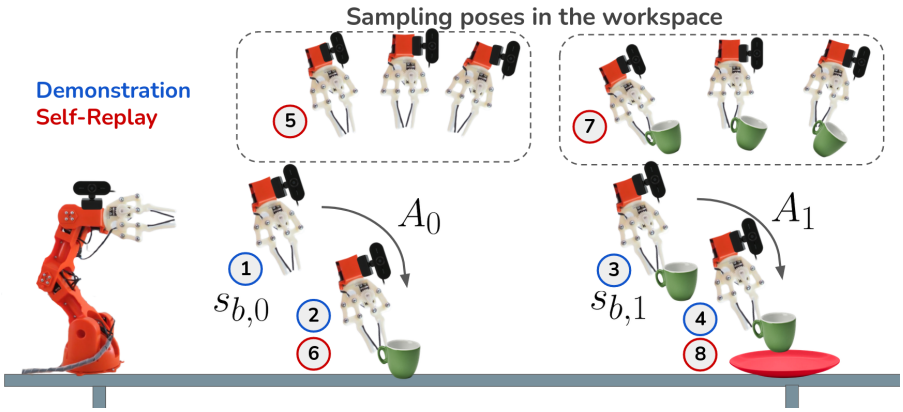


Figure 4.2: An example of the training steps for a two-stage task: picking a cup and placing it on a plate. The human operator brings the robot to the *bottleneck pose* on top of the cup (1), and records the object interaction phase by grasping the cup and lifting it (2). The operator then brings the robot on top of the plate (3), and records the second object interaction phase (4), by bringing down the cup and releasing it. The operator then resets the environment by placing back the cup. The Self-Replay phase starts: the robot gathers data to learn how to reach $s_{b,0}$ exploring the workspace (5). When the data collection phase ends, the robot moves to $s_{b,0}$ and replicates the first object interaction by executing the recorded actions A_0 (6). It then repeats these steps for the second stage (7,8). We suggest to watch the video on our website for a full example.

one-shot performance on single-stage tasks. However, it struggles with the multi-stage tasks we address, as error accumulation across stages degrades performance. Our Active Self-Replay extends these ideas with an active exploration strategy that focuses data collection on high-error regions of the state space.

4.3 Method

Background on Coarse-to-Fine Imitation Learning

One of the main concepts our method builds upon is *shifting the imitation learning problem* from learning a model that emulates the demonstration in its entirety, to learning to align the end-effector to the pose where the interaction demonstration started, so that it can repeat the exact actions performed by the operator. The end-effector therefore needs to reach the same relative position to the object that it had at the start of the demonstration. Following this framework, introduced in [110] as Coarse-to-Fine Imitation Learning, the human operator provides a demonstration composed of two phases (Fig. 4.1, 4.2): they first bring the robot to a pose close to the object, that is defined as the *bottleneck pose*. The bottleneck pose is the pose of the end-effector at which the interaction phase starts. Intuitively, we can imagine it as being fixed in the object's imaginary frame of reference, as it rigidly moves with it. From there, the

operator starts the interaction phase, in which they demonstrate how to precisely interact with the object, e.g. picking up a tool, inserting a a plug, etc. (Fig. 4.1). The assumption, proved effective in [110], is that if the robot can reach the bottleneck pose accurately, hence being in the same relative pose to the object as it was during the demonstration, then replicating exactly the actions recorded during the demonstration is sufficient to correctly interact with the object. The goal of the robot hence becomes to learn to reach the bottleneck pose accurately on novel configurations of the environment, i.e. novel poses of the objects composing the task, without the need to explicitly learn a policy to model the interaction phase.

The bottleneck pose is not an explicit part of the object or of its frame, but is chosen by the operator when providing the demonstration based on how they need to interact with the object. How does the human operator actually choose the bottleneck pose for each stage, separating the reaching and interaction phases? The bottleneck pose is arbitrary, and several choices are equally possible. However, as described in [110], during the interaction phase the actions are replicated in an open-loop manner, and external sources of noise can accumulate an error that is proportional to the length of the interaction phase. Hence, it is suggested to select a bottleneck pose that is sufficiently close to the object.

In a multi-stage task, the operator provides several interaction demonstrations, one for each stage (examples in Fig. 4.1, 4.2). Hence, we need to define one bottleneck pose per stage. We denote each bottleneck pose as $s_{b,N}$, where N denotes the N-th stage of the task, represented with the spatial coordinates x_b, y_b, z_b, θ_b, where θ is the rotation around the vertical, z-axis. We extract this pose from the robot's kinematics, effectively being a pose in 3D with rotation around the z axis, being it the pose where the demonstration of the interaction phase starts. More details are provided in following sections. We denote $s = \{x, y, z, \theta\}$ as a position and orientation of the end-effector. We define as o the observation, an RGB image that the robot receives from its wrist-mounted camera. Using a wrist-mounted camera, the observations are a function only of the relative pose between the end-effector and the objects. This is a fundamental aspect, as it allows the robot to autonomously gather data that can generalise to novel situations, as described in later sections. Each pose s is computed using the kinematics of the robot. At training time we also have the exact pose of the bottleneck $s_{b,N}$, extracted from the human demonstration, hence we can always reach it precisely through inverse kinematics. At test time this information is unknown and must be deducted from observations. Hence the robot has to learn how to map each RGB observations o_t, where t denotes the current time-step, to action a_t that moves the end-effector closer to $s_{b,N}$ in any novel configuration of the task it needs to solve. The actions are the end-effector spatial velocities, angular velocity around the

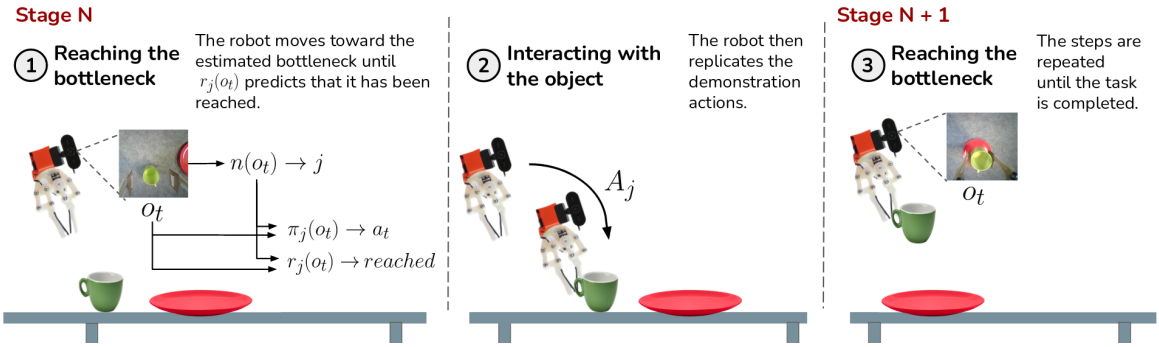


Figure 4.3: A visual description of the method solving novel configurations of a task at test time. Each stage is solved by first reaching the bottleneck pose, and then replaying the interaction actions. The image is passed through all the networks described in 4.3. The videos on our website provide additional examples of this.

z-axis, and a binary command to open or close the gripper, $v_x, v_y, v_z, v_\theta, c$. All the actions are given in the end-effector frame.

Tackling Multi-Stage Tasks: Overall Architecture

In the previous section we described how Coarse-to-Fine Imitation Learning [110] proposes to solve a novel configuration of a task for which it received a single demonstration: the end-effector must learn to reach the bottleneck pose, the relative pose it had with the object at the beginning of the operator demonstration, and then it can replay the actions executed during the demonstration. To solve a multi-stage task, we need to extend this method to alternate a series of bottleneck reaching and actions replay phases. To do this, we introduce a novel framework of neural networks and a novel self-supervised data collection algorithm, Self-Replay.

To solve a multi-stage task at test time, the robot needs to understand in which stage of the task it is at any time, how to reach the bottleneck of the predicted stage, and if that pose has been reached with sufficient precision, so it can start the interaction phase. All this information must be extracted from an observation, o_t in the form of an RGB image at each time-step t, as at test time the absolute pose of the bottleneck is not provided, nor the stage that the robot is currently in.

We designed a framework of three different networks to do so: first, a **stage-recognition network**, $n(o_t) \rightarrow j \in [0, N]$, that predicts the stage of the task the robot is currently in, in the form of a discrete class. This prediction conditions the remaining pipeline, as for each stage the bottleneck to reach is different. We predict the current stage at each time-step instead of simply starting from the first and shifting to the next after each interaction phase to be robust to failures and external disturbances. If, for example, the end-effector drops an object during a

stage, it needs to realise that it must go back to a previous stage to pick it up again. We then use a **bottleneck reaching network** $\pi(o_t) \rightarrow a_t$, that given the current RGB image captured from the wrist-mounted camera, predicts what action to take in the end-effector coordinate frame to reach the bottleneck pose, $s_{b,N}$. Finally, a **bottleneck classification network**, a network that predicts if the bottleneck pose has been reached, $r(o_t) \rightarrow \{0, 1\}$. If the network predicts that $s_{b,j}$ has been reached, i.e. is closer than a threshold, the robot executes the series of object interaction actions recorded during the demonstration for the j-th stage. The general pipeline is visually described in Fig. 4.3. Apart for the stage-recognition network, we train a different network for each task stage, and use the stage prediction j to select which one to use. We could have used a single network for each type, adding j to its inputs to condition its output, but we found training separate networks to result in better performance. In the following sections, we describe how Self-Replay can autonomously collect all the data needed to train the aforementioned networks, and hence learn to solve the multi-stage task, starting from a single demonstration.

Test Time Pipeline In Fig. 4.3 we visually describe the pipeline at test time, starting from an RGB observation o_t. The networks collaborate to predict (1) in which stage the robot is actually in, hence understanding what bottleneck pose should be reached (2) how the robot should move to approach the bottleneck of the predicted stage (3) if the bottleneck has been reached, and hence it is time to execute the interaction phase to complete the stage. If the bottleneck is considered non reached, the actions computed by $\pi(o_t)$ are executed by the robot, and then a new observation is obtained. All these estimations are computed from the same input observation o_t and recomputed independently at each time-step.

Providing the Demonstration

The human operator only needs to provide a single demonstration of the task, and then reset the environment as it was before the demonstration. No additional actions are needed from the operator. For each stage of the task, the human operator first brings the robot's end-effector to the object's bottleneck pose $s_{b,N}$ through free-space motion. In this phase, only the final bottleneck pose is recorded. While the choice is somewhat arbitrary, we found that positioning the end-effector 3-8cm above the grasp point (for grasping tasks) or 2-5cm from the contact point (for pushing/insertion tasks) works well across our task suite. They then demonstrate how to interact with the object moving the robot's end-effector. In this phase, all the actions commanded by the operator are recorded as $A_N = a_{0:T_N,N}$, where N denotes the N-th stage.

The operator then repeats these steps for all the stages composing the task. An example can be seen in Fig. 4.2.

The human operator then needs to reset the environment only one time as it was at the beginning of the demonstration, as the object interaction phases may have changed the initial configuration of the objects. This is in strong contrast with other approaches, like reinforcement learning-based methods, that require hundreds or thousands of resets [168], or classic behavioural cloning methods, in which the human operator needs to provide tens of demonstrations [217, 165], resetting the environment each time.

Collecting Data via Self-Replay

After providing a single demonstration of the full task and resetting the environment, bringing the objects to their starting position, all the following data collection and training phases are self-supervised, hence the human operator can disengage and no more human interventions are needed.

As described in Sec. 4.3, the agent needs to collect data to train a the models that will allow it to accurately reach the bottleneck pose of each stage from any environment configuration and starting pose. As described before, the use of a wrist-mounted camera makes the RGB observations a function of the relative, and not absolute, pose between end-effector and objects. Hence, moving the end-effector is generally equivalent to moving the object to observe it from different poses. To gather data in a self-supervised fashion we use a method we call Self-Replay.

Random Self-Replay Our algorithm extends the method proposed in [110] by being able to autonomously shift between stages to autonomously gather all the data it needs to solve the multi-stage task. We also propose a novel active variant in 4.3. The general approach is to sample a random pose of the end-effector in the workspace, and from there linearly go back to the bottleneck $s_{b,N}$, while recording the observations encountered in the path. The robot explores the area above the bottleneck, to avoid collisions with the objects, and we assume that area to be free from obstacles. More precisely, at each time-step the agent stores data in the form $(o_t, s_{e,t}, s_{b,N}, N)$, where o_t is the image taken from the camera in that pose, $s_e = x_e, y_e, z_e, \theta_e$ is the pose of the end-effector at the current time-step, $s_{b,N}$ is the bottleneck pose, and N denotes the stage of the task we're currently in as an integer. For each pose we can accurately compute the ground-truth relative displacement between the current end-effector pose and the bottleneck, and therefore analytically compute what end-effector movement would bring it closer to the bottleneck. This allows us to then train a model to predict the optimal movement

from an RGB observation. This method of data gathering is defined Random Self-Replay, as we randomly sample poses in the workspace to explore different relative poses between end-effector and objects. While randomly sampling poses would be enough, we propose a more efficient and robust active exploration strategy, called Active Self-Replay, that we describe in Sec. 4.3. When the data collection of a task stage is completed, the robot needs to shift to the next stage. To do so, it reaches the bottleneck pose $s_{b,N}$, and then replicates the recorded actions A_N to interact with the object (e.g. moving an object, grabbing an object, etc.). After this phase, the algorithm goes to the next task stage $N \leftarrow N+1$, repeating the aforementioned steps (Fig. 4.2), with the next bottleneck pose $s_{b,N+1}$. By autonomously moving between stages, the robot can gather all the data it needs for all the stages without any additional human intervention.

Active Self-Replay We additionally propose an active, self-supervised data collection method. While both methods alternate between collecting data in a self-supervised fashion and then replaying the demonstration's actions to shift to the next stage, the steps of data collection are different. The first stage is identical to the Random Self-Replay phase described before. It collects data repeating the Random Self-Replay cycle M times, gathering an initial dataset. However, differently from Random Self-Replay, we then train our models on this data also during the training phase: after having trained the networks on this initial dataset, the robot will then sample poses from its workspace that maximise the error of the networks, hence areas that have not been explored enough and require more training data. This allows the robot to actively collect more informative data, instead of re-sampling already explored areas. We use a gradient-free sampling based optimiser.

The data collection and network training steps are repeated until the robot is unable to find a pose in its workspace where the network's error is larger than a threshold δ_{err}, or after a maximum number of steps defined beforehand.

4.4 Experiments

We evaluated our proposed method on a set of real-world, everyday multi-stage manipulation tasks. Based on these experiments, we now answer the following questions: (1) Is our method able to solve a series of common, multi-stage tasks with a single demonstration? (2) Is the Active Self-Replay algorithm more efficient and robust than the Random variant? (3) How does our method compare to similar state-of-the-art methods from the literature? (4) Is reproducing the actions executed by the human operator during object interaction enough to solve the task, or

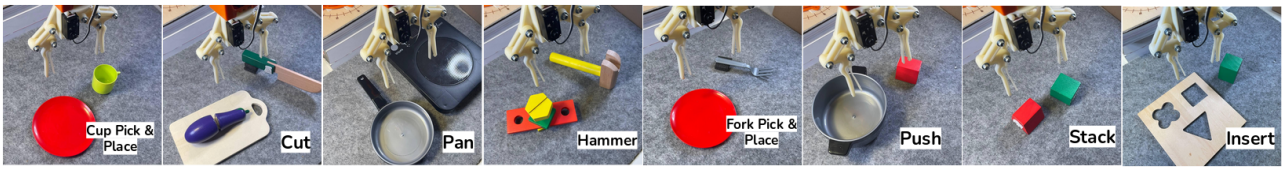


Figure 4.4: The set of tasks we tested our method on.

would a learnt policy perform better?

Experimental Setting and Tasks

Due to the governmental restrictions during the COVID-19 pandemic, we were unable to conduct experiments in our institution's laboratory. Therefore, we decided to set up a robotics laboratory at home, building a smaller, 6-DOF manipulator with a parallel gripper: the TinkerKit Braccio. Despite the small size of this robot, we were able to conduct all the experiments successfully and answer the above questions, albeit with a reduced workspace. Nevertheless, the results obtained here can effortlessly be replicated on a larger robot. We use a wrist-mounted RGB camera, that captures 800×600 pixels images, which we resize to 64×64. We control the robot through a simple keyboard interface that maps keypresses to end-effector delta positions commands. In particular, we use 6 keys to move along the 3 spatial directions of $\pm 0.5\texttt{cm}$, 2 keys to rotate along the z axis of ± 10 degrees, and one key to open and close the gripper. Our setup is lightweight and easy to reproduce, only requiring a wrist-mounted RGB camera with no calibration needed. All the networks in this work are convolutional neural networks trained with supervised learning on a single GPU.

Manipulation Tasks We designed a set of 8 tasks, inspired by the recent imitation learning literature [110, 45, 218, 217, 67, 165], and by everyday tasks to test our method. We included a variety of common multi-stage human tasks, like picking-and-placing, cutting food, shape insertion, etc. (Fig. 4.4). To generate a test set for the experiments, we instantiate novel configurations of each task by moving the objects in the workspace. Position are sampled inside the $30cm \times 20cm$ workspace, and orientation is sampled in a 120 degrees range. The end-effector starts 15 cm above the table. In all the tasks, the robot end-effector can move in 3D space and only rotate around the vertical z-axis. For all experiments, task success is evaluated by the human operator at the end of each episode through binary judgement (success/failure). We define success criteria as follows:

- Cup (Pick-and-Place): The cup must be lifted from its initial position, transported, and

placed stably on the target plate without falling.

- Fork: As above, but with the fork being the subject object.
- Stack: All specified cubes must be placed in a vertical stack that remains stable for 3 seconds after the robot releases the final cube, place on top of the target spot on the table.
- Cut: The knife must be grasped and used to perform a cutting motion through the toy vegetable, achieving visible separation.
- Insert: The object must be inserted into the corresponding hole.
- Push: The red object is placed in the pot and the latter is pushed by at least 3cm to the side.
- Pan: The pan is lifted and placed on the stove.
- Hammer: The hammer must be grasped and used to touch the target.

Network Architectures

All the neural networks we implemented in this work are Convolutional Neural Networks. In particular, we use Deep Spatial Autoencoders (DSAE) [65], that autonomously extract keypoints from RGB observations as a form of information bottleneck. This resulted in a strong regularisation during the training phase.

The inputs are 64×64 RGB images, from which we extract from 4 to 16 keypoints based on the task. For all the networks described in this work, we train a DSAE with an L2 reconstruction loss in image space, to learn to extract keypoints in the form of (x_i, y_i). We the give the array of K keypoints as an input to a feed-forward network to predict the actions, the current stage, or the proximity to the bottleneck pose.

Results

Learning Multi-Stage Tasks from a Single Demonstration We tested our method by providing a single demonstration of each task. Recording the demonstration takes roughly a minute to the human operator. A video detailing the experiments can be found on our website. We compared Active Self-Replay and Random Self-Replay, (Sec. 4.3) (Table 4.1). Both variants

Table 4.1: Percentage of successes of the variants of our method, Random Self-Replay (R. S-R) and Active Self-Replay (A. S-R) and a series of baselines on 20 configurations of each tasks. Experiments are repeated over 3 random seeds and show mean and 1-std.

10*[-1em]	**Task**							
Method	Cup	Fork	Stack	Cut	Insert	Push	Pan	Ham.
FlowC.	40 ± 4.1	24 ± 2.3	13 ± 9	5 ± 4.1	6 ± 2.3	13 ± 2.3	12 ± 2.3	18 ± 2.3
SIFT	8 ± 2.3	0 ± 0	0 ± 0	0 ± 0	0 ± 0	0 ± 0	0 ± 0	3 ± 2.3
LMP-1	3 ± 2.3	0 ± 0	0 ± 0	0 ± 0	0 ± 0	0 ± 0	0 ± 0	0 ± 0
RIL-1	0 ± 0	0 ± 0	0 ± 0	0 ± 0	0 ± 0	3 ± 2.3	2 ± 2.3	2 ± 2.3
LMP-20	62 ± 2.3	48 ± 6.2	38 ± 6.2	23 ± 4.7	20 ± 4.0	33 ± 2.3	50 ± 7.0	$53 \pm 10.$
RIL-20	60 ± 4.0	52 ± 4.7	38 ± 4.7	17 ± 2.3	18 ± 2.3	37 ± 2.3	48 ± 6.2	52 ± 2.3
DDPGfD	3 ± 2.3	$0 \pm 0.$	$0 \pm 0.$	$0 \pm 0.$	$0 \pm 0.$	$0 \pm 0.$	$0 \pm 0.$	7 ± 2.3
BC-20	30 ± 6.2	25 ± 4.0	12 ± 2.3	7 ± 4.7	8 ± 2.3	25 ± 7.0	28 ± 2.3	$42 \pm 10.$
R. S-R	83 ± 4.7	63 ± 4.7	58 ± 2.3	57 ± 2.3	77 ± 4.7	63 ± 4.7	62 ± 6.2	62 ± 2.3
A. S-R	88 ± 2.3	83 ± 2.3	73 ± 4.7	73 ± 4.1	73 ± 4.7	77 ± 4.7	73 ± 4.7	73 ± 4.7

are trained on the same amount of self-collected data, which is collected autonomously by the robot, following the methods described in 4.3. We then used the data to train the networks described in Sec. 4.3, keeping the architecture, number of parameters and random weights initialisation fixed on both Active and Random variants. We then set up a test set of 20 random initial configurations of each task by placing the objects randomly in the environment, testing both methods on this same test set and averaging the results over 3 random seeds. At test time, the robot behaves following the method we described in Sec. 4.3 and Fig. 4.3. In our experiments we observed how, although Random Self-Replay achieves strong performance, it failed on some particularly challenging configurations of the harder tasks, e.g. when the difference in orientation of the object with respect to what observed during the demonstration was large. Indeed, we observe improvements of Active over Random Self-Replay on the harder tasks, like *Cut*, that requires precise orientation of the knife, and *Insert*, which also requires precise orienting the object above the shape (Table 4.1).

Comparison with Baselines from the Literature. We also compared Self-Replay to a series of techniques from the literature, to highlight the improvements of our method. We divide the baselines we developed and compared into three categories:

1 - One-Demonstration Methods (most comparable to ours): FlowControl [13]: Uses a pre-trained optical flow network (RAFT) to compute dense pixel correspondences between the current observation and the demonstration bottleneck image. These correspondences are converted to an alignment action via least-squares fitting. Key difference from Self-Replay: FlowControl uses a fixed, pre-trained feature extractor that was not designed for robotics, while our method learns task-specific visual representations through self-supervised data collection. FlowControl also cannot handle multi-stage tasks without modification. SIFT Matching [128]:

Uses classical hand-designed SIFT features for correspondence matching, followed by the same alignment procedure as FlowControl. Key difference: SIFT features can often try to match parts of the background which are texture-rich but meaningless for the task, while our learning based method learns to ignore such details.

2 - Multi-Stage Behavioural Cloning: Latent Motor Plans (LMP) [132]: Learns a latent space of motor primitives from demonstrations using a variational autoencoder, enabling generation of multi-stage behaviours by planning in latent space. Key difference: LMP requires many demonstrations (20+) covering the full range of task variations, as it must learn the complete behaviour distribution. Self-Replay instead decomposes the problem, requiring only a single demonstration of the interaction phase. Relay Imitation Learning (RIL) [82]: Specifically designed for multi-stage tasks, RIL learns a goal-conditioned policy that can chain sub-tasks by using the final state of one sub-task as the goal for the next. Key difference: RIL requires demonstrations of the full task from many initial configurations (we test with 10 and 30), while Self-Replay uses a single demonstration plus autonomous data collection.

3 - Reinforcement Learning from Demonstrations: DDPGfD [194, 146]: Uses demonstrations to initialise a DDPG policy and the replay buffer, then continues learning through environment interaction. Key difference: DDPGfD requires continuous human supervision for environment resets and reward shaping, whereas Self-Replay requires no human involvement after the initial demonstration and reset. Additionally, DDPGfD struggles with sparse rewards in multi-stage tasks.

We compare our results on a series of everyday-like tasks in Table 4.1. We provide 1 and 20 demos to the behavioural cloning methods. We provide 1 demonstration to DDPGfD, as the human operator also has to spend a considerable amount of time supervising the autonomous exploration phase and resetting the environment. We show that all the baselines fail when receiving only a single demonstration. LMP and RIL perform better than BC, while DDPGfD is unable to solve these tasks consistently. Here, we show that our method is not only more time efficient for the human operator, but can also achieve better performance when using the same number of datapoints than state-of-the-art baselines. Those experiments additionally show that RIL and LMP require around 40-50 demonstrations to reach the performance we obtain with a single demonstration, hence being 40x to 50x more time expensive than Self-Replay.

Table 4.2: Rate of successes and 95% confidence Wilson intervals of our method against an end-to-end behavioural cloning method, Relay Imitation Learning, with 10 or 30 demonstrations, over 20 configurations of each task. We measure how performance changes as the number of stages of a task increases.

	Stages		
Method	1	2	4
Ours	1. [0.84, 1.00]	0.9 [0.70, 0.97]	0.8 [0.58, 0.92]
RIL-10	0.6 [0.39, 0.78]	0.4 [0.22, 0.61]	0.15 [0.05, 0.36]
RIL-30	0.9 [0.70, 0.97]	0.75 [0.53, 0.89]	0.35 [0.18, 0.57]

Comparison with Behavioural Cloning on an Increasing Number of Stages

We designed a set of experiments to compare the efficiency and performance of our method on tasks with an increasing number of stages while still receiving only a single demonstration, with respect to behavioural cloning (BC), where instead the human operator provides several demonstrations of the task. In particular, we compare it to Relay Imitation Learning (RIL) [82], a recent variation that extends behavioural cloning to multi-stage tasks. We designed a stacking task (Fig. 4.4), where the robot has to stack a varying number of cubes in the correct order, where we can easily add more stages by using more objects.

In our method, we only provided a single demonstration, running Active Self-Replay to autonomously gather data. We compared it to RIL with 10 and 30 demonstrations. For RIL, we provided end-to-end demos of the multi-stage task from 10 or 30 different initial configurations of the environment, recording RGB observations and actions in the end-effector frame in the form of $\{(o_0, a_0), \ldots, (o_T, a_T)\}$. We then trained a policy network f_{BC} with supervised learning on these demonstrations to map $f_{BC}(o_t) \rightarrow a_t$. We used the same network architecture we use as our bottleneck reaching network (Sec. 4.3).

In Table 4.2, 1 stage corresponds to only grasping the first object, 2 stages to also placing it on a goal x, y position on the table, 4 stages to also place the second object on top of the first. We test the two methods on the same test set of 10 configurations of the task, measuring the number of successes. We show how behavioural cloning, extended to tackle multi-stage tasks with the Relay Imitation Learning method, quickly starts to degrade in performance when the number of stages increases, while our method always obtains a large number of successes.

Comparing Time and Sample Efficiency of Self-Replay against Baselines

We designed our method to minimise the amount of human effort needed to teach a robot everyday-like manipulation skills. In fact, our method only needs one demonstration from the human operator and one environment reset to learn autonomously how to solve the task also when facing novel initial configurations, as demonstrated in Section 4.4. In this section, we show how time efficient, and also sample efficient, is our method compared to the baselines presented in Section 4.4. When talking about time efficiency, we refer to the time that the human operator needs to spend in any phase of the training, either providing demonstrations or resetting the environment. When talking about sample efficiency, we refer to the number of datapoints needed to learn to solve the task by each method. Despite our method is able to autonomously gather large datasets without the need for human supervision or intervention, we still show how the Coarse to Fine decomposition makes our method more sample efficient than the baselines.

We compare Active Self-Replay against other state-of-the-art imitation learning baselines introduced in 4.4, Latent Motor Plans (LMP) and Relay Imitation Learning (RIL). We recorded the time needed to provide demonstrations to each method, and tested their performance on a set of test configurations of the task over time. Our method only receives a single demonstration and does not require any further supervision, hence the time needed to train it is a couple of minutes of operator time. We also compared to Deep Deterministic Policy Gradient from Demonstrations (DDPGfD). We provided one demonstration and then supervised the autonomous reinforcement learning phase, by both providing rewards and resetting the environment. We tested the performance of the method over time as well.

In Figure 4.6, we show how our method can achieve the best performance on all the tasks, while also requiring an order or more of magnitude less time from the human operator. We also tested the sample efficiency of the various methods, by training them on the same amount of datapoints, gathered either by Self-Replay, demonstrations, or autonomous exploration in reinforcement learning. As we show in Figure 4.7, our method obtains a better sample efficiency than the others. This is mostly due to the Coarse to Fine structure of our method: the policy network only needs to learn how to align the end-effector to the bottleneck pose, moving in free space, without the need to learn the fine manipulation skill, e.g. cutting, stacking, picking objects. By replaying the expert's demonstration, we allow the policy to only focus on the accurate alignment.

These experiments additionally show that RIL and LMP require around 40-50 demonstrations to reach the performance we obtain with a single demonstration, hence being 40x to 50x more time expensive than Self-Replay.

Reinforcement Learning from Demonstrations We tested a recent state-of-the-art reinforcement learning from demonstration technique: Deep Deterministic Policy Gradient from Demonstrations (DDPGfD) [146, 194]. We provided a demonstration to bootstrap the exploration phase of the algorithm. We use a sparse reward as in [146], provided by the expert to the algorithm at training time, rewarding the robot if the task is successfully completed. The operator hence needs to spend time recording the demonstrations and supervising the learning phase both for providing the rewards and resetting the environment after each training episode. Engineering an automatic system to reliably provide the reward would have also required a considerable engineering effort. Even with an order of magnitude more operator's time spent on supervising the learning phase with respect to the operator's time needed in our method, DDPGfD was not able to solve even the simpler tasks, as also observed in [110]. As showed in [194], these methods can require even million of time-steps, and are better suited to simulators, where environment resets are automatic. Detailed results can be found in Table 4.1.

Robustness to Distractors

While our method gathers data autonomously by only observing the objects composing the task, we empirically investigated the performance of the method to solve the task at test-time in the presence of additional, unseen objects. Furthermore, we move the lighting source during testing, to additionally test the method's robustness to visual distractors. This considerably changes the visual aspect of the environment, as can be seen in Figure 4.8. The keypoints based neural architecture we use, described in 6.1, autonomously learns to track the objects of interest with keypoints. This generates a strong regularizing effect: as we show in Figure 4.8 and in the videos on our website, the keypoints appear to be largely unperturbed by the presence of distractors, as they are trained to only extract relevant features to the task. We included videos of tasks with visual distractors on our website that better illustrates the method's robustness.

Table 4.3: Rate of successes in the presence of visual distractors in the environment. We also include average success rate with 95% confidence Wilson intervals. These results show the presence of distractor does not strongly impact the performance of our method.

	Task				
Distr.	Cup	Cut	Fork	Hammer	Average
Without	0.85	0.65	0.80	0.70	0.75 [0.64, 0.83]
With	0.80	0.50	0.70	0.65	0.65 [0.55, 0.76]

We empirically tested the performance of our method to solve tasks in the presence of distractors. As the results in Table 4.3 show, the task solving performance does not drop considerably, hence showing robustness to unseen objects in the environment. We train our method as described in Section 4.4, only using the objects relevant to the task. We then setup 20 different configurations of the task by moving the objects on the table, and we additionally randomly place a set of distractors.

Replicating the Interaction Trajectory vs. Learning an Interaction Policy

In the recent robot learning literature, policies are often times learnt from data using a function approximator, like a neural network. We designed a set of experiments to show that learning a policy for the interaction phase from demonstrations does not bring any observable benefits with respect to replaying the operator actions during the interaction phase, while requiring more human effort. We provided one demonstration for each task and ran Active Self-Replay to train our framework of networks (Sec. 4.3). We additionally trained a policy network f_{BC} with 10 human demonstrations using behavioural cloning, collected starting from poses sampled close to the bottleneck pose and then recording RGB observations and actions in the end-effector frame in the form of $\{(o_0, a_0), \ldots, (o_T, a_T)\}$ to map $f_{BC}(o_t) \rightarrow a_t$.

We tested both methods on a test of 10 novel configurations of four of our tasks (Table 4.4). Since we're only interested in comparing the interaction phase, we used the same bottleneck reaching network trained with Active Self-Replay on each test configuration to reach the bottleneck. When the robot predicted that the bottleneck had been reached, we recorded the current pose of the robot (which may also differ from the ground-truth bottleneck pose, as it is the result of the estimations of the networks) and ran the interaction phase replaying the actions seen during the demonstration. To then test the policy, we reset the pose and the objects as they were before the interaction. We then executed the interaction by predicting

Table 4.4: Rate of successes of replicating the operator's actions during the interaction phase against using a policy learnt from demonstrations. We also include average success rates with 95% confidence Wilson intervals. These results show that our replay method, using 1 demo, achieves the same performance as a policy trained with 10, when starting from the pose reached after the alignment phase.

	Task				
Method	P&P	Cut	Insert	Stack	Average
Ours	0.90	0.70	0.80	0.80	0.80 [0.65, 0.89]
Policy	0.90	0.80	0.70	0.80	0.80 [0.65, 0.89]

actions from RGB observations o_t using the trained policy network f_{BC}.

During the experiments, we observed that, often, the end-effector was able to precisely reach the bottleneck pose, thanks to the experience gathered with Self-Replay. Therefore, replicating the recorded actions was sufficient to successfully solve the task, and the learnt policy did not increase the robustness of the method. On particularly challenging configurations of the objects, the alignment was not precise, but this resulted in both methods failing. Results therefore show that learning the interaction phase from demonstrations does not provide any statistically significant benefit (Table 4.4).

4.5 Limitations

Despite the ability to learn multi-stage tasks with a single demo, this method is limited to a certain family of tasks: mostly tasks where the relative positions of objects between each other is not influential, and that do not require precise, dexterous closed-loop adaptation while interacting with the object. Furthermore, the method is limited to top-down interaction during the bottleneck reaching phase. Better self-supervised data collection strategies could allow for 6 degrees of freedom bottleneck reaching.

4.6 Discussion

In this work, we presented a novel method to learn to solve multi-stage, everyday-like manipulation tasks from a single human demonstration. We empirically demonstrated how our method can solve a wide range of tasks, often used in the literature as benchmarks, while requiring minimal human work, no prior knowledge, engineering, or modelling of the objects to manipulate. In the broader context of this thesis, this work is an effort in designing methods that, for a certain category of tasks, can let the robots collect their own data. This can help

tackle the need for large datasets of paired observations and actions, especially in parts of the task where the robot is mostly performing free space movement.

In this chapter, we described a method that can learn tasks in isolation, and we did not explore the challenge of learning a repertoire of tasks, or generalising to new objects. In the next chapter, we will extend the presented method with additional techniques to tackle the aforementioned challenges.

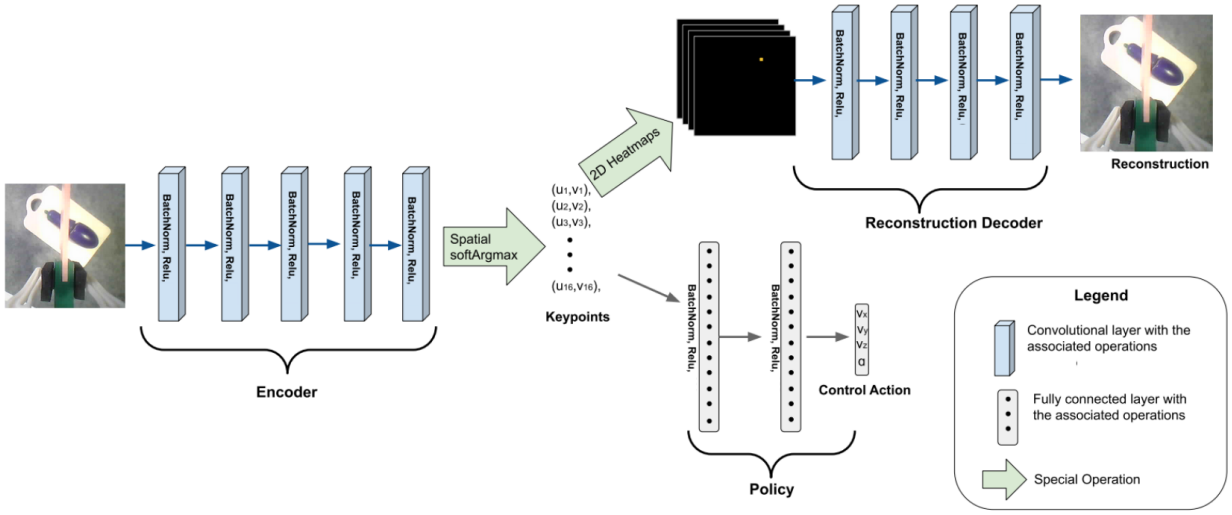


Figure 4.5: General neural network architecture used in this work, inspired by [191, 65]. We show in this figure the policy network π. The other networks have an almost identical structure, with the only difference being the last layer of the fully-connected network.

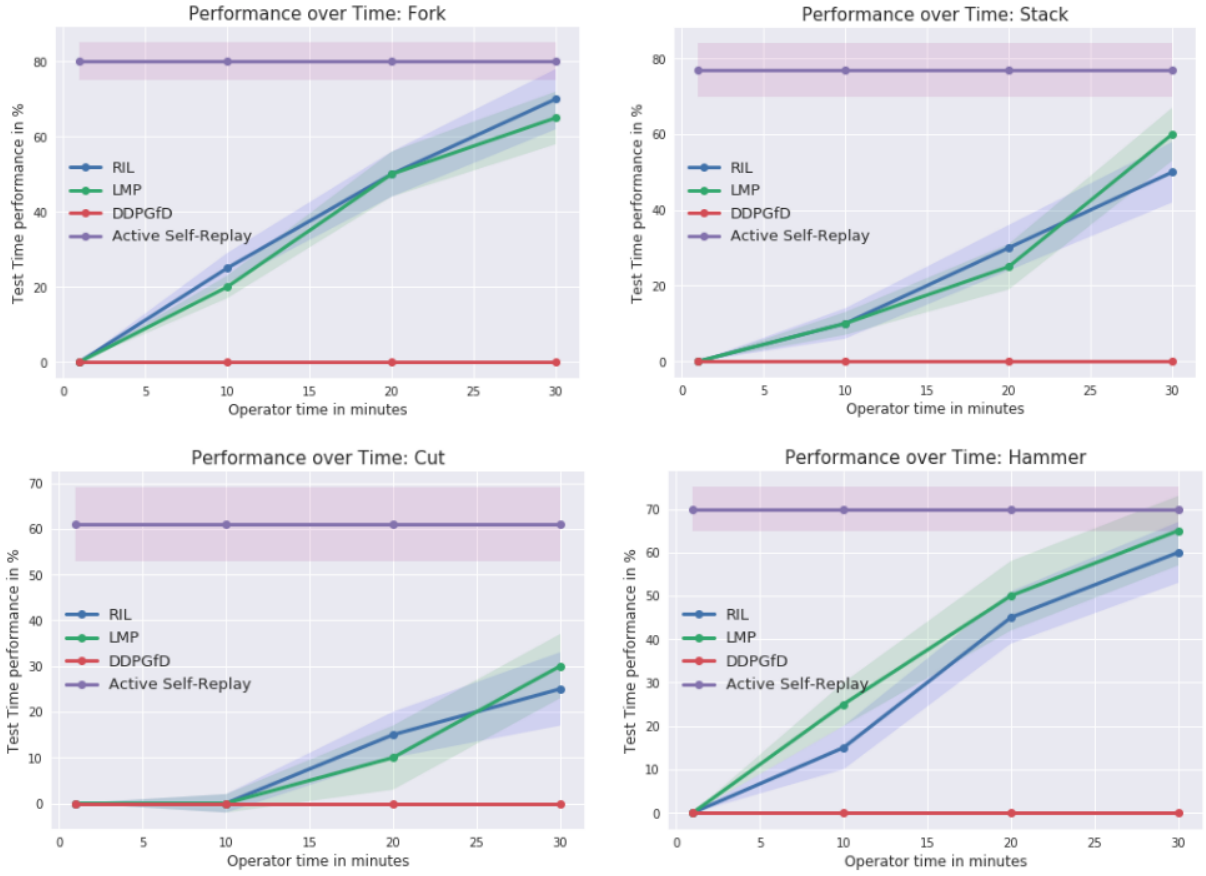


Figure 4.6: In these plots, we compare the time efficiency against task solving performance of Self-Replay and various baselines.

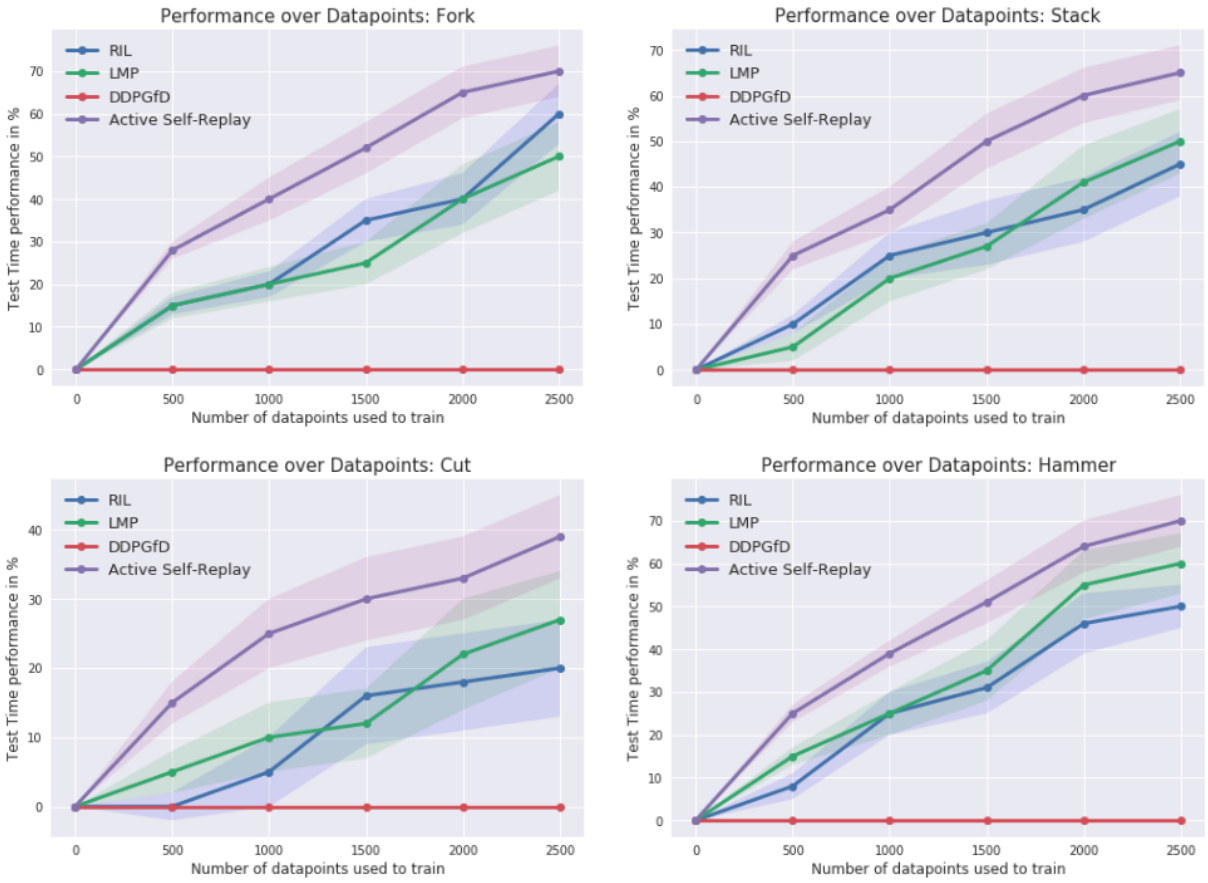


Figure 4.7: In these plots, we compare the sample efficiency against task solving performance of Self-Replay and various baselines.

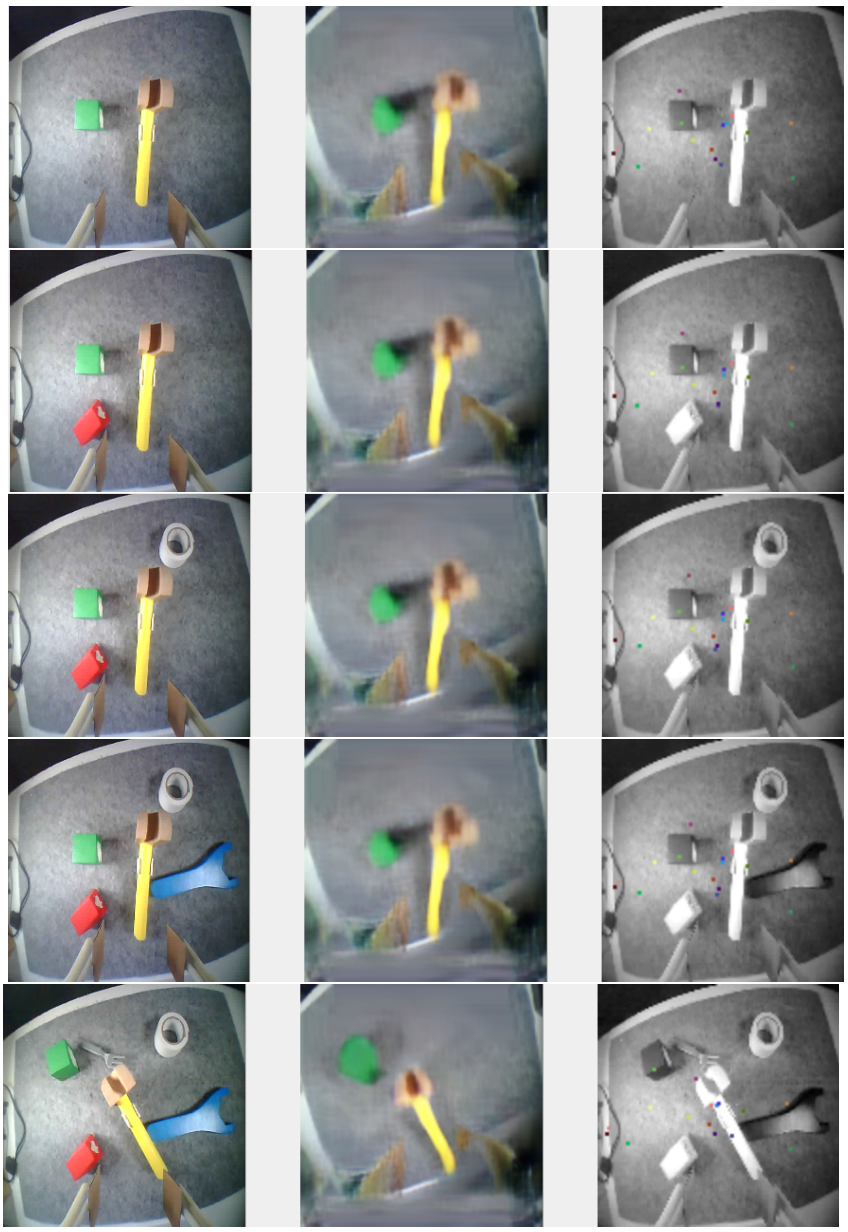


Figure 4.8: An example of the robustness of our method to visual distractors. From left to right: original image, network reconstruction, and visualisation of the keypoints autonomously learnt by the network. The robot is trained to pick up the hammer and hit the green target, and the keypoints-based Deep Spatial Autoencoder learns to focus on those objects. By adding distractors (one more for each row), the reconstruction stay mostly unchanged, showing that the network is not confused by the unseen objects. In the last row, me move the task's objects, and can see how the reconstruction is able to follow that movement, hence showcasing how the network is able to predict the objects' poses. It can also be seen that the keypoints remain mostly unchanged: this shows that the network output is not influenced by distractors, and becomes a function only of the task's objects. We strongly suggest to watch the video on our website that better visualizes how the keypoints learn to track only the task's objects. Additional videos on our website demonstrate the method robustness to these visual distractors.

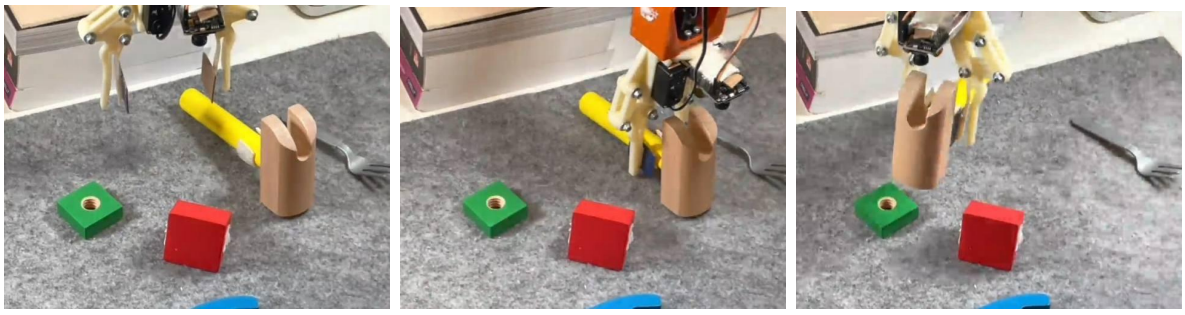


Figure 4.9: An example of the Hammer task with additional distractor objects. Additional videos on our website demonstrate the method robustness to these visual distractors.

Chapter 5

On the Effectiveness of Retrieval, Alignment, and Replay in Manipulation

5.1 Introduction

In this chapter, we study the problem of teaching a robot how to interact with a set of training objects, and then generalising these learnt behaviours to novel objects and novel classes of objects, extending the ideas introduced in the previous chapter. Today's dominant paradigm in the recent literature is to address this with end-to-end behavioural cloning [40]. However, to generalise everyday manipulation skills to many different objects, such techniques require a very large number of human demonstrations, which is slow and expensive.

But as an alternative to monolithic, end-to-end control, we can decompose reasoning into three distinct, specialist modes of reasoning. Firstly, *what* can a robot do with an object? Secondly, *where* should a robot interact with an object? And thirdly, *how* should a robot interact with an object? Our hypothesis is that this decomposition might be more effective than expecting a single control policy to be able to reason simultaneously about all three modes. In our previous chapter, we focused mostly on the last two scenarios, assuming the robot would learn tasks in isolation and therefore did not have to understand what to do with a new, possibly unseen object at hand.

To achieve these three modes of reasoning, we propose a new framework based on three principles: **retrieval, alignment, and replay**. When provided with an object during testing, these are then used as follows. First, to determine *what* can be done with the object, the most

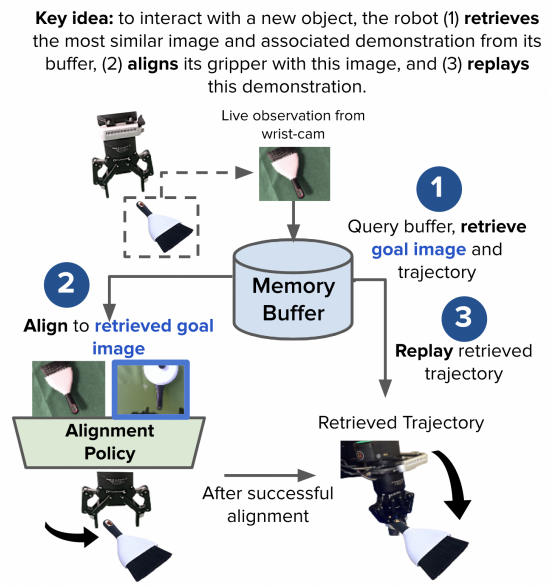


Figure 5.1: An overview of the framework we study, showing the *retrieval*, *alignment*, and *replay* phases. Together, these enable one-shot imitation learning without prior object knowledge, as well as generalisation to novel objects and novel classes.

visually similar training object to the observed object is *retrieved* from a memory buffer of the training objects. Second, to determine *where* the robot should interact with the object, a goal image representing the retrieved training object is used in a goal-conditioned *alignment* policy, to align the end-effector appropriately with the object. Third, to determine *how* the robot should interact with object, the robot *replays* the demonstration velocities for that training object.

As such, our hypothesis is that, in the absence of an explicit demonstration for a novel test object, interacting in the same way as with the most visually similar training object, might be sufficient to interact with the test object. And through a series of real-world experiments, we demonstrate this to be true for a range of everyday tasks such as grasping, pouring, and inserting. To empirically validate our hypothesis, we design a large set of experiments comparing our proposed solution and decomposition with a series of methods from the recent literature that only adopt one of our proposed modes of reasoning. In the Experiments section, we will discuss in more depth how each experiment will help answer a particular hypothesis about our proposed method. We study whether our proposed decomposition is optimal compared to methods which do decomposition but not retrieval (e.g. [121]), to methods which do retrieval but not decomposition (e.g. [155]), and to methods which do neither decomposition nor retrieval (e.g. behavioural cloning [40]). Our conclusion, and the key takeaway message, is clear: decomposing into retrieval, alignment, and replay leads to an order of magnitude better learning efficiency compared to all these other alternative paradigms. Furthermore, **our framework only re-**

quires a wrist-mounted camera, without the need for either extrinsic external or intrinsic calibration, and begins learning *tabula rasa*, with no previous knowledge of the objects required, such as CAD models: our only assumption is that the relative pose between wrist-camera and end-effector remains fixed between train and test time, i.e. the camera is rigidly mounted to the robot.

5.2 Related Work

Many recent methods that learn object manipulation from demonstrations, learn end-to-end control policies using behavioural cloning, which can be conditioned on a task identity or language command to enable generalisation to novel objects. However, such policies tend to require extensive datasets of human demonstrations [40]. In our work, we show that an explicit decomposition into *retrieval*, *alignment*, and *replay*, results in a substantial improvement in data and time efficiency over end-to-end control methods.

Our work builds on top of Coarse-to-Fine Imitation Learning (C2F-IL) [110], which also proposes alignment and replay. However, C2F-IL learns tasks in isolation: it learns a single control policy for a single object, which does not generalise to novel objects. As such, we study a framework based on retrieval, which can then scale up and learn from many different demonstrations across many different objects, generalising to both novel objects and novel object classes, and becoming more and more capable with each new demonstration.

Other works have proposed a decomposition of the robot trajectory into a coarse, approaching stage and a fine, last-inch interaction [121, 209, 25, 202, 198]. However, most have stronger constraints in their design compared to our proposed framework: [209] needs synthetic visual data, and therefore models of the objects, to train a perception system in simulation. Our method starts learning *tabula rasa*, and needs no prior knowledge of the tasks or objects, therefore making it substantially more general and deployable in everyday environments. [25] also performs trajectory decomposition, but needs external, calibrated RGB-D cameras to identify affordances. [200] frames robot manipulation entirely as a series of objects alignments, it however relies on point clouds and calibrated external cameras. Our method only needs a single uncalibrated wrist-camera, which is significantly easier to deploy in unstructured environments.

The use of retrieval for few-shot learning, especially in robotics, reinforcement learning and control, has also been investigated elsewhere [53, 155]. [155] directly retrieves the actions to execute, whilst [53] retrieves task-related data from an unlabelled buffer to train an end-to-end policy. We combine retrieval and decomposition in a novel way, through an interplay of

retrieving goal observations and trajectories to execute and learning of an alignment policy through self-supervised learning.

Our retrieval and replay framework can be viewed through the lens of offline reinforcement learning (offline RL) [124], which learns policies from fixed datasets without environment interaction. A key capability of offline RL methods like Decision Transformer [34] and Conservative Q-Learning [117] is "trajectory stitching"—composing novel behaviours by combining segments from different trajectories in the dataset. In our framework, retrieval serves a similar function: by selecting demonstrations based on visual similarity to the current observation, we implicitly "stitch" together relevant experience from different demonstrations. However, our approach differs in several important ways: 1) 2e decompose tasks into alignment (learned) and interaction (retrieved) phases, while offline RL methods must learn when and how to switch between trajectory segments. 2) our retrieval operates in visual feature space, enabling generalisation to visually similar but physically different objects. Offline RL typically operates in state space, requiring demonstrations that cover the relevant state distribution. Future work could combine our visual retrieval mechanism with offline RL trajectory stitching. For example, rather than replaying a single retrieved trajectory, we could use offline RL to plan over multiple retrieved trajectory segments, potentially enabling more complex tasks that require combining skills in novel ways. This would require addressing challenges in (1) defining appropriate reward functions for retrieved segments and (2) handling the distribution shift between training and test conditions.

5.3 Method

We aim to provide robots with the capability to both efficiently learn to interact with an object, and transfer the learnt behaviour to novel objects. To achieve this, we propose a novel imitation learning framework based on an interplay between **retrieval** from a memory buffer of visual observations, a learnt goal-conditioned visual **alignment** policy, and **replay** of demonstrated trajectories. Our method only needs an RGB-D wrist-camera, rigidly mounted to the robot's end-effector, without the need for any calibration. No prior knowledge of objects or tasks is needed, and no external camera is needed.

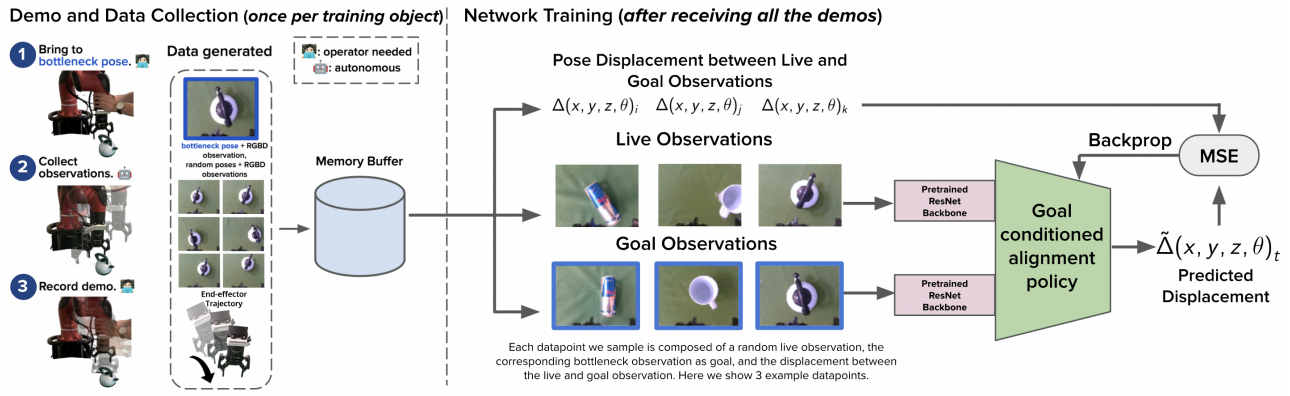


Figure 5.2: We collect a single demonstration per object following the steps in Sec. 5.3. Through autonomously collected data, our method trains a goal-conditioned alignment policy, that will align the robot to the bottleneck pose before executing the trajectory. All collected data is stored in the memory buffer.

Preliminaries

Unless explicitly stated otherwise, we use these terms with the following meanings: a *trajectory* executed by the robot is composed of 3D linear and 3D angular velocities, expressed in the end-effector frame. A *pose* refers to a 4D vector x, y, z, θ_z expressed in the world frame: it can describe either the pose of an object on a table, or the pose of the end-effector in the environment. A *policy* is a function mapping visual inputs to action outputs. An *action* is an end-effector velocity executed at each time-step. A *task*, in this work, can be intuitively described as a verb, like *grasp*. The same task can be performed on many objects (e.g. grasping a mug, grasping a teapot). The environment is a tabletop setting with an object the robot should interact with. All the tasks we consider involve executing full 6-DoF velocities during object interaction.

An Overview of the Retrieval, Alignment and Replay Phases

In this section, we provide a high-level overview of our framework and its workflow, as illustrated in Figure 5.1.

When deployed during testing, the robot starts by obtaining a visual observation of the environment with its wrist-camera. To understand *what* to do with the object it has observed, it queries an external memory buffer with the visual observation. The memory buffer **retrieves** *where* to interact with the object in the form of a goal visual observation, and *how* to interact with the object as a trajectory, expressed in the end-effector's frame. This retrieved data is used to guide the robot's behaviour as follows.

The goal of the robot, after retrieving this information from the buffer, is simple: to move its end-effector so that the live observation is **aligned** with the retrieved goal observation. This is done via visual servoing through a learnt, goal-conditioned visual alignment policy. Once the alignment is completed, the robot **replays** the retrieved trajectory by executing the trajectory of end-effector velocities. This allows for successful interaction with new objects, under the hypothesis that objects which are visually similar should be interacted with in the same way, which we later validate experimentally.

The following subsections describe in more detail how each part is implemented.

Understanding the Retrieval, Alignment, and Replay Phases

The order in which the phases are executed during deployment is *retireval*, followed by *alignment*, followed by *replay*. However, for clarity, we will now adopt the reverse order to describe them. This will allow us to first start with simple concepts, and to later describe the overall framework, providing the reader with a more intuitive sequence of ideas. The *alignment* and *replay* phases are inspired by Coarse-to-Fine Imitation Learning [110], also described when introducing Self-Replay in the previous chapter, which we now build upon using *retrieval* to allow for generalisation to novel objects, and scalability by allowing for data from multiple demonstrations and across multiple tasks to be absorbed by the robot.

The Replay Phase: Recording and replaying demonstrated trajectories

To teach the robot how to interact with an object, the human operator manoeuvres the end-effector to provide a demonstration, e.g. with kinesthetic teaching. The end-effector starts from a pose b^O, chosen by the operator and expressed in the imaginary object frame O, that we call the *bottleneck pose*. The only requirement is that the object must be visible by the wrist-camera from the bottleneck pose. The demonstration is recorded as a series of 3D linear and 3D angular velocities of the end-effector expressed in the end-effector frame E, which we call a *trajectory* $s = [\mathcal{V}_1^E, \ldots, \mathcal{V}_T^E]$, where $\mathcal{V} = [v_x, v_y, v_z, w_x, w_y, w_z]$.

We can imagine the bottleneck pose being rigidly attached to the imaginary object frame. With W being the world frame, when we move the object, b^W moves rigidly with it, while b^O stays constant. If the end-effector is re-aligned to b^W when the object is moved, re-executing the end-effector velocities of the recorded trajectory would suffice in solving the task again [110]. Therefore, the learning process can focus on reaching b^W again for novel poses of the object. The *replay* phase is hence the execution of a recorded trajectory. We store all the

trajectories demonstrated during training in our memory buffer as $[s_1, \ldots, s_D]$, where D is the total number of demonstrations.

The Alignment Phase: Learning to align to the bottleneck again

Before the demonstration starts, the operator aligns the end-effector to b^O, which is wherever the operator plans to start recording the trajectory from. In this configuration, the robot can compute the exact pose of b^W through forward kinematics. The goal of the *alignment* phase is reaching b^W again at test time, when its location cannot be simply computed from forward kinematics (because the object has now moved), and thus must be predicted from visual observations.

Our approach is to train a goal-conditioned visual alignment network that, guided by the wrist-camera observations, aligns the end-effector to b^W. The network receives, as input, the live observation of the wrist-camera, and an observation recorded at b^W: its goal is to align the former to the latter by moving the end-effector, and hence, the wrist-camera. To train this neural network, we need a dataset of visual observations of the object from many different poses, and the relative pose displacement that would align the end-effector to b^W from each pose.

We gather this dataset as follows. We store the *bottleneck visual observation* o_b taken from b^W and later use it as the goal observation when training the network. The robot then moves in the volume above b^W, reaching new random poses p_i^W and collecting visual observations o_i from its wrist-camera. Knowing both p_i^W and b^W through forward kinematics, it can compute the pose displacement d_i^E in end-effector frame E that would bring the end-effector back to b^W, aligning o_i to o_b. All this data is added to the memory buffer. Once this autonomous process is complete, the operator can manually manoeuvre the end-effector to provide the demonstration. As shown in [110], this enables a task to be learnt from just a single demonstration. This process is illustrated in Figure 5.2, left side.

Once all the desired D demonstrations have been collected, across a number of objects and tasks, our method can train the goal-conditioned visual alignment network through regression, using the data stored in the memory buffer. As illustrated in Figure 5.2 right side, we sample pairs of o_b, o_i collected from the same object, along with the pose displacement d_i^E. We use the former as inputs to the network, and the latter as the label, optimising the weights through backpropagation and gradient descent using an MSE loss.

At test time, when the object is in a different configuration and b^W is unknown, we feed the

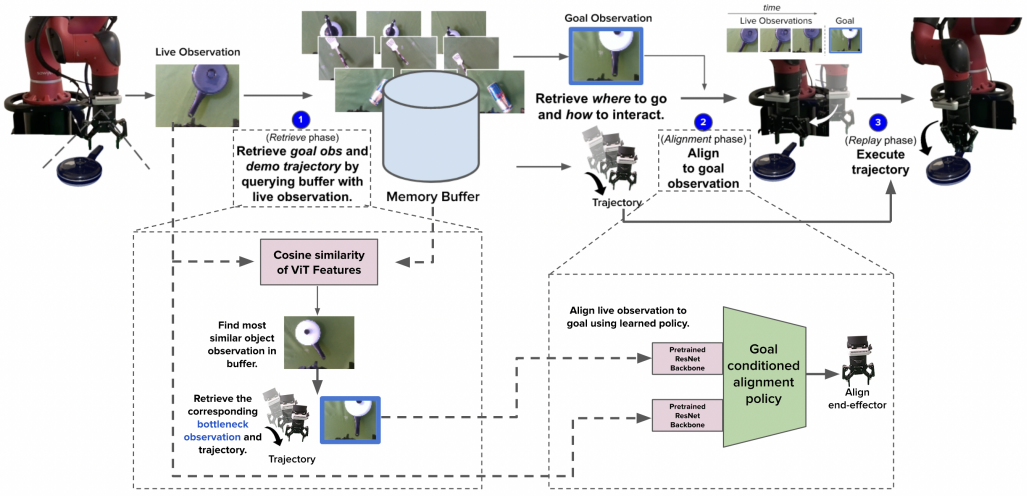


Figure 5.3: Overall illustration of our framework. Upon observing a new object, the agent visually compares it to other objects it observed during training. After finding the most similar object, the agent retrieves a goal observation of that object to guide the visual alignment phase (*where* to interact), and also a trajectory to execute once the alignment is complete (*how* to interact).

visual input from the wrist-camera to the goal-conditioned visual alignment model along with the goal observation to align to, which will then predict the pose displacement d_t^E between p_t^W, the current pose of the end-effector, and b^W: moving the end-effector by that pose will align it to b^W, so that the *replay* phase can then be executed. This process constitutes the *alignment* phase.

But, when at test time, and the robot observes a novel object, how does the robot obtain a goal observation to guide the goal-conditioned visual alignment network for the *alignment* phase? Similarly, how does the robot then decide which trajectory to execute during the *replay* phase? The robot must autonomously obtain this information guided only by the visual inputs it receives. This constitutes the *retrieval* phase, as now described.

The Retrieval Phase: Goal image and trajectory retrieval from the memory buffer

After the training stage, where the robot receives a demonstration per each object in the dataset, our robot is equipped with an external memory buffer which stores the following: 1) the D trajectories recorded by the operator, 2) the D bottleneck visual observations $o_{b,d}$ used as goal observations, and 3) the observations $o_{i,d}$ collected for each object at random poses by the wrist-camera, with $0 \leq d < D, d \in \mathbb{N}$, $0 \leq i < I, i \in \mathbb{N}$, I being the total number of observations collected per object, and $o_{i,d}$ therefore being the i-th observation of the d-th object.

After receiving the visual observation of an object the robot should interact with, the goal

of the *retrieval* phase is twofold: 1) to retrieve one of the D *bottleneck visual observations* to use as the goal for the goal-conditioned visual alignment network, guiding the *alignment* phase, and 2) to retrieve one of the D trajectories that, if executed during the *replay* phase, will lead to a correct interaction with the object. The visual aspect of the novel object should therefore guide the choice of *where* to align with the object, and *how* to interact with it.

The robot obtains this information by querying the memory buffer with the live observation. Our framework uses a similarity function $g(o_{live}, o_{i,d}) \rightarrow [-1, 1]$ computed between the current live observation and all observations stored in the buffer. It firstly computes $o_{j,n} = \text{argmax}_{o_{i,d}} g(o_{live}, o_{i,d})$, and then, it retrieves both the corresponding *bottleneck visual observation* $o_{b,n}$, and the corresponding trajectory s_n. Intuitively, replicating the alignment and interaction of the most visually similar object in the buffer studies our hypothesis that *visually similar objects should be interacted with in the same way.*

The recent computer vision literature proposes a variety of techniques to extract useful representations from visual inputs to map them to smaller dimensional embedding vectors, which capture semantic information of the image, where it is more efficient and effective to compute distances using well known mathematical functions. In our case, the similarity function is computed as the cosine similarity between the visual features of the RGB channels of the two observations extracted from a Vision Transformer (ViT) [51] trained using DINO [31]. The literature has demonstrated how these features capture both geometric and semantic information on both a local and global scale, generalising between similar objects of the same or different classes [10]. We extract these features for all the observations in the buffer offline, after the training stage. During deployment, we extract features of the live observation and compute the cosine similarity with all the stored features in parallel. This process is illustrated in Figure 5.3.

As we will show in our experiments, retrieval from the buffer unlocks effective generalisation, while decomposition of object manipulation into alignment and replay brings a substantial improvement in learning efficiency. While the latter has been explored in [110, 192] and the previous Chapter, the combination of retrieval and decomposition is distinctly novel, and leads to a general and scalable framework that can continue to improve with further data. Furthermore, with this framework, teaching a task on an object takes *no more than a minute of human time*, and this is an important property when considering practical, real-world imitation learning. The human operator simply provides just a single demonstration per object; the remainder of the data collection (the visual observations) is autonomous, and training is therefore self-supervised.

5.4 Experiments

In this section, we empirically measure the ability of our *retrieval, alignment, replay* framework to learn behaviours efficiently and transfer them effectively to new objects. We use four everyday tasks: **grasping** an object, **pouring** from a cup into a container, **unscrewing** a bottle's cap, **inserting** a bottle's cap, and **inserting** toy bread in a toy toaster (we combine these two inserting tasks in one group). These tasks require non-trivial 6-DoF interactions with objects, and some require considerable precision (inserting a bottle's cap allows for a maximum of 4mm of error). We invite the reader to watch the videos on our website at https://www.robot-learning.uk/retrieval-alignment-replay.

Setup: We run our experiments on a Sawyer robot with 7 degrees of freedom, with a Robotiq 2F-85 gripper. The robot is controlled via position control during the alignment phase and velocity control during the interaction phase. We use a wrist-mounted RGBD Intel RealSense camera only, which does not need any calibration in our method. Our only assumption is that the end-effector to camera extrinsics are constant between training and testing, i.e. the camera remains fixed, a common assumption when mounting a wrist-camera to a robot [25, 46, 110]. The camera receives RGBD images which we rescale to $128 \times 128 \times 4$. We use a set of everyday objects (Figure 5.5) to illustrate the method's ability to tackle common tasks. The agent starts learning *tabula rasa*: **no previous knowledge of tasks or objects, such as CAD models, is used**. Each method we study is trained with demonstrations using the training set objects depicted in Fig. 5.5, top bar plot. Each is then tested using the objects depicted in all three bar plots: training objects (top), intra-class unseen objects (middle), inter-class unseen objects (bottom).We use precise terminology to describe our generalisation experiments: **Object category (or class):** A semantic grouping of objects with similar function and general appearance, e.g., "mugs," "bottles," "shoes." Objects within a category share a common interaction affordance (e.g., all mugs can be grasped by their handles) but may differ in specific geometry, colour, and size. **Training objects:** Specific object instances used during the demonstration collection phase. **Novel objects (test set):** Object instances not seen during training. We distinguish two types:

1) Intra-class novel objects: New instances from the same category as training objects (e.g., a different mug). These test semantic generalisation—can the robot recognise that a new mug should be manipulated similarly to known mugs? **2)** Inter-class novel objects: Objects from entirely new categories not present in training. These test broader transfer capabilities. We

curated our object set to satisfy several criteria relevant to real-world robot deployment:

Diversity within categories: For each category (mugs, bottles, shoes, etc.), we selected objects with variation in size, shape, and appearance (different colours, materials, textures). Objects were sourced from online marketplaces, and our laboratory's existing inventory. **Task relevance**: We selected object categories that appear frequently in prior robot learning benchmarks [[82]; [110]; [40]] to enable comparison with prior work. Categories include: containers (mugs, cups, bowls), tools (kettles, pans), insertables (caps, bread), and miscellaneous household items. **Practical graspability**: All objects can be grasped by a standard parallel-jaw gripper (Robotiq 2F-85). For this reason, many of our objects are toy versions of real objects (toy pans, toy toasters) as their reduced size make them easier to interact with.

The wrist camera, rigidly mounted to the end-effector, starts the episode at 70cm above the table, and its field-of-view can observe the whole table in front of the robot, as demonstrated in a video on our website.

Baselines

Different methods in the recent literature have proposed augmenting robot manipulation with retrieval or decomposition of the trajectory, similar to the approach we adopt [121, 209, 25, 155]. We investigate the contribution to manipulation performance brought by these building blocks, to motivate our design choices and better understand the effect of each.

In particular, we identify 4 different families of approaches, grouped by whether or not they perform retrieval from a buffer of data gathered during demonstrations, and whether or not they decompose object manipulation into an alignment part followed by an interaction part, or treat it as a single learnt trajectory. A visual taxonomy of these families is provided in Figure 5.4. For each of those families, we implement and evaluate a method from the recent literature. To have a fair comparison, we chose methods that require similar assumptions to our method: as we discuss in Section 5.2, some methods require either numerous, precisely calibrated cameras, and/or 3D models of the objects, while ours requires neither.

Our method is composed of both a *retrieval* phase and a decomposition into *alignment* and *replay*. GUAPO [121] also decomposes object manipulation into a coarse *alignment* phase, approaching the object by feeding visual observations into a neural network, and a last-inch learnt policy is used to perform the interaction. This method lacks an explicit *retrieval* phase, as the affordances are recognised implicitly by a neural network. To train this method, we use the same steps described in 5.3 to record data and train an alignment visual servoing

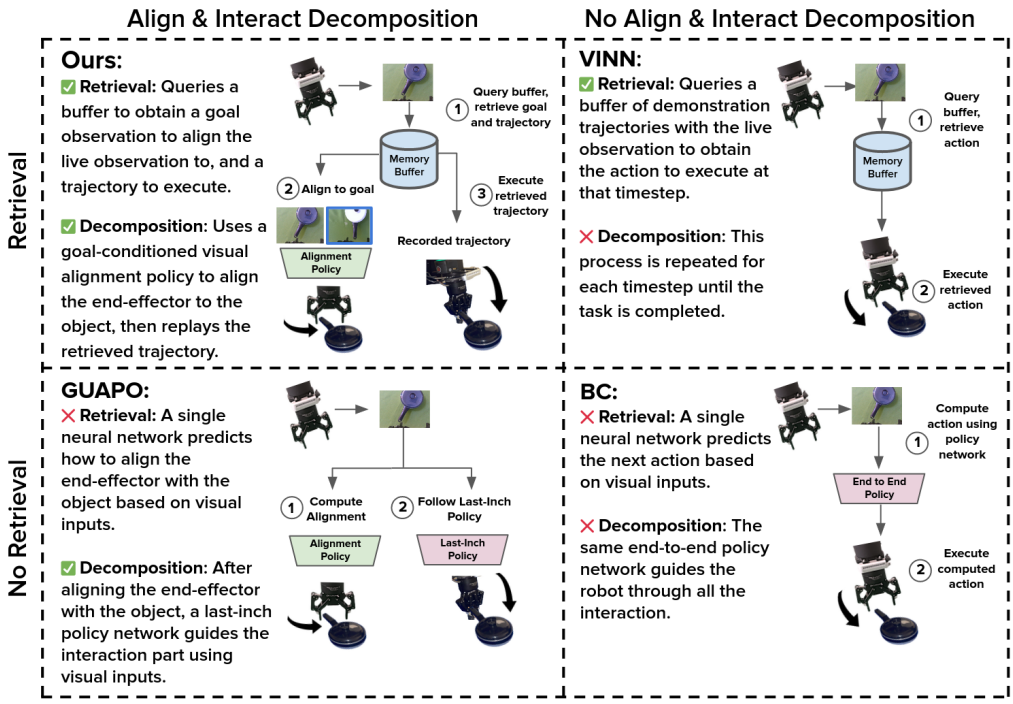


Figure 5.4: A taxonomy of the different robot manipulation paradigms we study and examples from recent literature, based on the concepts of retrieval and/or decomposition.

network, which in this case is not goal conditioned, but only receives the live observation as an input. Differently from our method, it uses a last-inch policy network that we train through behaviour cloning (BC) to interact with the object. As the original paper [121] proposes the use of reinforcement learning for the policy, while we use behaviour cloning for time efficiency, we denote this method as BC-GUAPO.

Visual Imitation through Nearest Neighbours (VINN) [155], frames learning from demonstration entirely as a retrieval problem. Demonstrations are stored frame by frame in a memory buffer, together with the action taken by the expert at that timestep. At each timestep during deployment, the agent queries the buffer with the current observation to find the most similar k observations. The action to execute is computed as a weighted average of the actions corresponding to these k retrieved observations. While it is retrieval based, this method does not decompose object manipulation trajectories, and instead models manipulation as a single trajectory.

Finally, behaviour cloning [40] is implemented as a baseline that does not use retrieval nor decomposition, but trains a single policy network with the data collected during the demonstrations, and executes its actions at deployment guided by visual observations. In this case, we train an end-to-end policy network through behaviour cloning using the collected demonstrations.

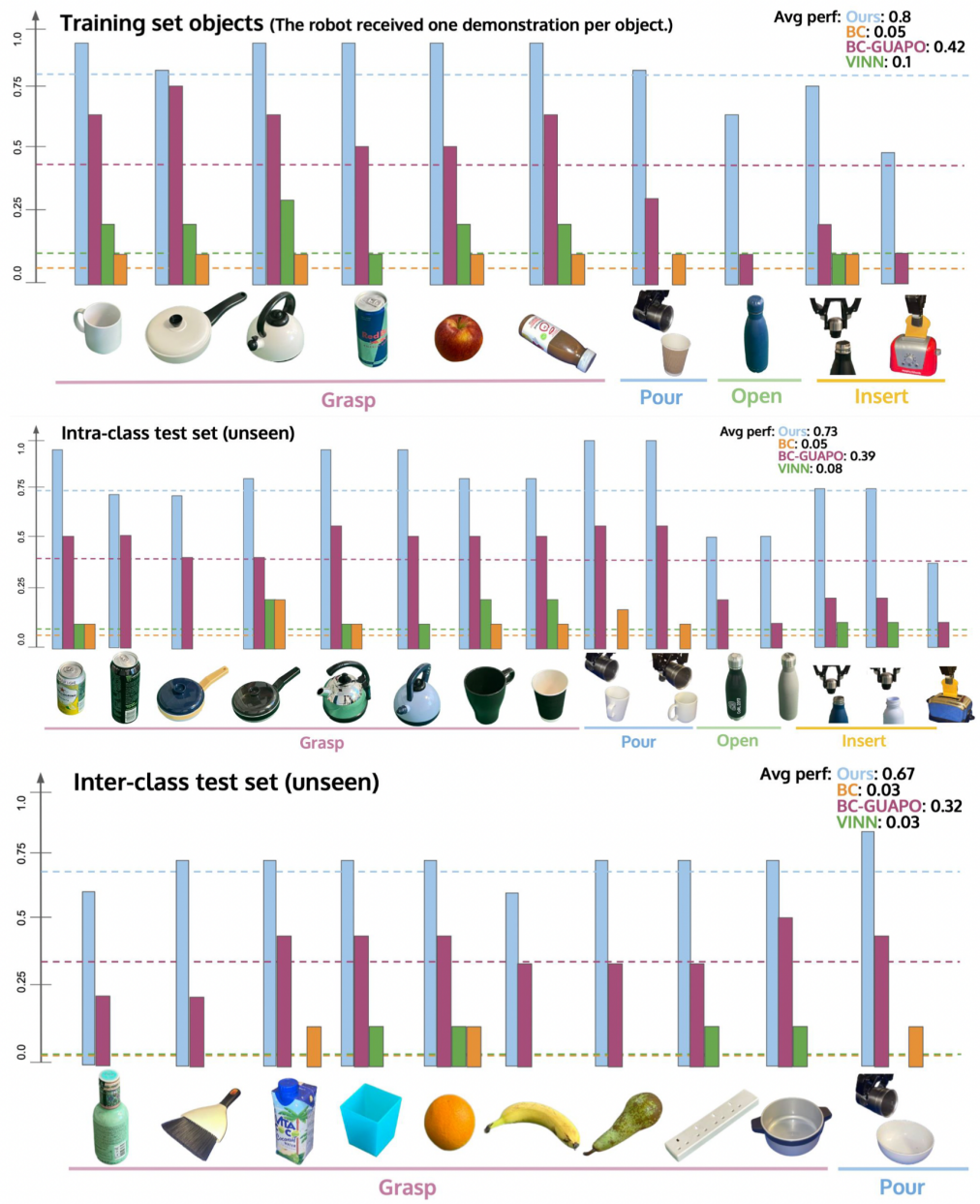


Figure 5.5: Performance of our method and baselines on a series of manipulation tasks involving tens of different objects, with the average performances as a dashed line.

We optimised neural network architectures and hyperparameters independently for each method, in an attempt to maximise the performance of each and obtain a fair comparison across these paradigms. To improve generalisation to visually similar objects, all the alignment or policy network we train for each method extracts visual features from the RGB channels through a pretrained, frozen ResNet-50 backbone [89], and then concatenates the depth channel, scaled to be in the same range as the extracted features.

Can our framework learn everyday-like manipulation behaviour efficiently, and transfer those trajectories to novel objects? What contribution do retrieval and decomposition bring, and how does this paradigm compare against recent baselines

from the literature? To answer this, we train all baselines providing demonstrations of how to interact with the objects in the training set (Fig. 5.5, top). As our method requires a single demonstration per training object, in our first experiments we provide a single demonstration per training object to all the baselines as well.

We cannot measure inter-class performance on the *bottle cap insertion*, *unscrewing a bottle cap* and *inserting toy bread in a toaster* tasks, as these tasks require a particular class of target object: we therefore only test for inter-class generalisation on the *grasping* and *pouring* tasks, that can instead be performed on a larger variety of unseen object classes (e.g. pouring can generalise from mugs to bowls). At test time, we present each object from both the training set and the test set to the robot. For each testing episode, the object is randomly positioned on the table in front of the robot. Through retrieval, the robot must understand how to interact with it based on the demonstrations it received (i.e. what task to perform: grasp it, open it, etc.). For example, if the robot observes a toaster and the bread slice in its end-effector, it should infer that the task is to *insert the bread*. It must then correctly execute the task, hence measuring its ability to generalise to novel poses of the objects, and also to novel objects. We run 10 trials per object, sampling a new (x, y, z, θ_z) pose each time. **We train on 10 objects and test on 35 objects, for a total of 350 test-time episodes for each baseline. Of these 35 objects, 25 were novel objects, and 10 belonged to completely novel classes, demonstrating strong generalisation and adaptation abilities.**

Results for our method's performance on each single object can be seen in Fig. 5.5. Here we show the success rate, and group those results into training set objects (for which the robot received demonstrations), intra-class objects (unseen objects from the same categories of the training objects, e.g. mugs), and finally inter-class objects (unseen objects from novel categories). All these results are obtained without the need for either intrinsic nor extrinsic camera calibration, and without any form of prior object knowledge.

The results indicate not only that our method can obtain remarkable performance with just a single demonstration per object (an average of 80% success on training objects after receiving a single demonstration for each), but that it can generalise to intra and inter-class objects: the average performance stays considerably close to the training set performance. (73% success percentage on the intra-class set with respect training set, and 67% on the inter-class set). The strong performance of our method on unseen, inter-class objects demonstrates that the proposed pipeline is robust even when there is not a clear object to retrieve from the buffer. When asked to grasp a banana, our method understands that the most similar object is an horizontally-placed bottle by matching geometrical features, as both share a locally cylindrical-

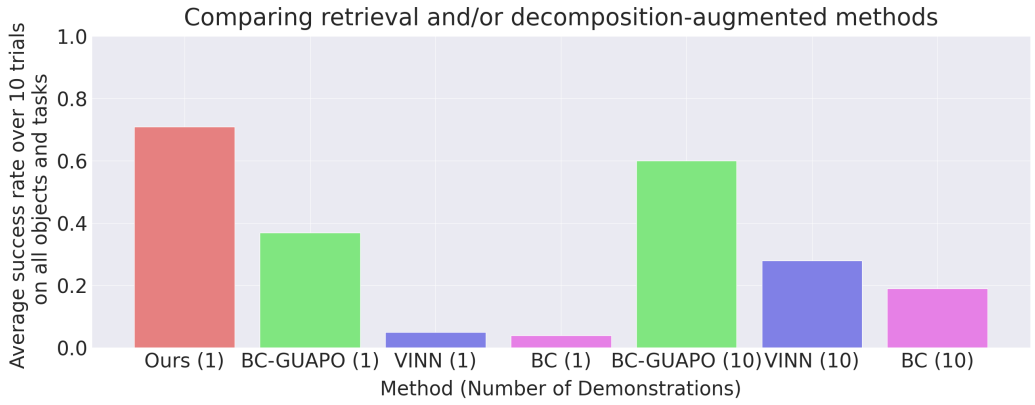


Figure 5.6: Average success rate of our method and the baselines, where each is provided with either (1) or (10) demos per training object.

like shape, as we show in the videos on our website. While performance on unseen, inter-class objects is lower than on training or inter-class testing objects, we can observe a much less steep decrease compared to baselines that do not use retrieval and/or alignment (Fig. 5.5).

Can we measure the performance benefits brought by retrieval and alignment both in isolation and then combined? We compare the overall performance of all methods, and how each scales with more data, in order to more closely study the individual contributions of the retrieval component, and the decomposition component (into alignment and replay). Figure 5.6 shows the results of these experiments, where each method is provided with either 1 demonstration or 10 demonstrations per training object, and where we measure the average success rate over all the objects and tasks of both the training set and the intra and inter-class test sets, running 10 episodes per object.

Comparing these results with the taxonomy proposed in Figure 5.4 allows us to clearly identify the contribution brought by the align/interact decomposition and/or retrieval. Behaviour cloning uses neither of those, performs the worst of all the baselines whether 1 demonstration or 10 demonstrations per object are provided. VINN is based exclusively on retrieval, and when receiving a single demonstration per object performs poorly as well. This suggests that retrieval alone is not sufficient to enable efficiency and effective transfer. On the other hand, BC-GUAPO, which decomposes trajectories into two phases similar to our method, performs much better than VINN with just a single demonstration per object. This clearly indicates the benefits of decomposing manipulation in this way, because the coarse alignment phase can be trained in a self-supervised manner from just a single demonstration.

When comparing BC-GUAPO to our method, however, we see a substantial gap in performance: as our method is based on both retrieval and decomposition, this suggests that *retrieval brings a noticeable performance boost when paired with decomposition.* In particular,

from our experiments it emerges that retrieval benefits from a large memory buffer of observations: when receiving 10 demonstrations per object, VINN clearly surpasses BC as well due to VINN benefitting from the use of retrieval. Our framework combines the best of both worlds: the decomposition into *alignment* and *replay* enables the robot to autonomously gather a large buffer of demonstrations without the need for human supervision. Given this buffer, the use of *retrieval* then enables the robot to generalise efficiently to novel objects.

5.5 Limitations

Currently, our method relies on visual inputs to predict what to do in an environment, and assumes the presence of a single object at test time: multiple objects may therefore lead to ambiguity. However, if we use an off-the-shelf object segmentation method, we can in principle interact with each object in succession. Future work will address these limitations. Like the method in the previous chapter, this method is limited to top-down bottleneck reaching. This is not a fundamental limitation of our method; if required, the alignment phase could be extended to 6-DoF by simply collecting images over 6-DoF end-effector poses. Also, in our tabletop implementation, although the alignment phase is 4-DoF, the replay phase is in fact 6-DoF, enabling complex interactions. Furthermore, like the method in the previous chapter, this method cannot be applied to tasks that require complex, dexterous, closed-loop interaction with objects in its current form, but in this work we show that a large set of everyday-tasks can be tackled with our framework, and that it is surprisingly competitive and often better in terms of efficiency and performance compared to learnt closed-loop policies.

5.6 Discussion

In this work, we propose a novel framework to enable efficient and effective learning from demonstration in robotic manipulation. In particular, we introduce a combined use of *retrieval*, and trajectory decomposition into *alignment* and *replay*. We analysed the best way of implementing retrieval and decomposition by comparing baselines from the literature, and showed that our combination of retrieval, alignment, and replay, performs better than alternative paradigms. Furthermore, we demonstrated how retrieving and replaying demonstration trajectories is a simple yet powerful alternative to training end-to-end policies, with significant gains in data efficiency.

To tackle the need for data in robot learning, in these last two chapters we proposed meth-

ods that allow robots to gather data in a self-supervised fashion, and train network policies from scratch. However, this still requires a considerable amount of robot time. Over the years, the field of computer vision saw the emergence of large, general vision networks trained on enormous, task-agnostic datasets of web-scraped images [30, 152]. These networks demonstrated outstanding vision understanding capabilities, and quickly became the backbone for many downstream vision tasks, avoiding the need to retrain networks from scratch, and instead unlocking either zero-shot abilities or efficient fine-tuning. In the next chapter, we will explore how the emergence of these networks can allow us to skip the data collection process, and build policies based on the strenghts of these vision networks.

Chapter 6

DINOBot: Robot Manipulation via Retrieval and Alignment with Vision Foundation Models

6.1 Introduction

As we mentioned in the first two chapters of this thesis, the recent major successes in deep learning all had two main common ingredients: enormous, web-scale datasets, and substantial computational power to train increasingly large neural networks. However no such dataset of comparable size is available for robotics, where sensory perception must be coupled with actions.

To take advantage of these large, pre-trained networks, the robotics community has often used them as backbone representations, to then train a neural network to predict actions on top of the extracted representations. However, imitation learning (IL) using these foundation models often still requires a considerable number of demonstrations for generalisation to emerge [147, 179, 164, 177].

We argue that, rather than integrating foundation models into existing imitation learning methods as a backbone representation, we can design new imitation learning frameworks around these new capabilities of foundation models. To this end, we introduce **DINOBot**, a new imitation learning framework for robot manipulation tasks, which leverages the key capabilities of Vision Transformers (ViTs) trained through DINO [31] (which we call DINO-ViTs), a self-supervised method for training vision networks.

As described by DINO's authors, DINO-ViTs extract "*universal features suitable for image-*

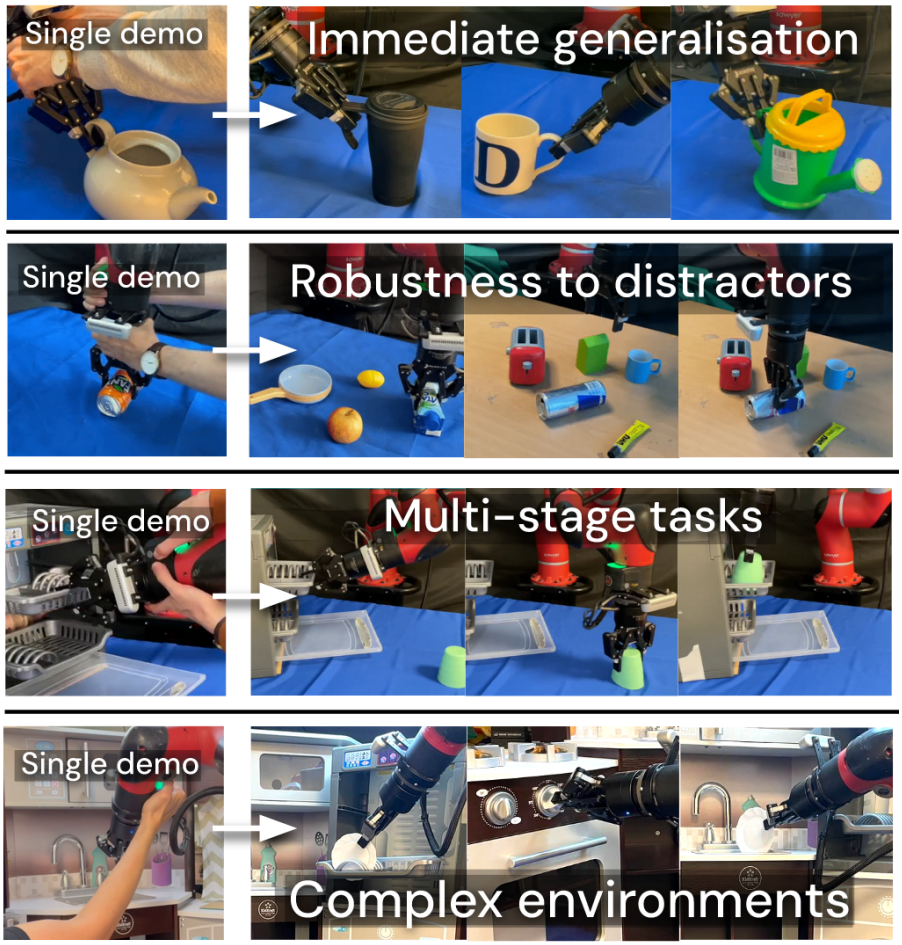


Figure 6.1: From a single demo, DINOBot can learn to adapt to new objects, be robust to distractors, execute multi-stage long horizon tasks, and interact with complex environments.

level visual tasks as well as pixel-level visual tasks" [152]. Inspired by these two capabilities, we designed DINOBot around two distinct modes of reasoning. Firstly, image-level *semantic* reasoning to generalise learnt behaviours to novel objects. Secondly, pixel-level *geometric* reasoning to generalise learnt behaviours to novel object poses. We integrate these two modes by modelling a manipulation task as a semantic image *retrieval* task, followed by a geometric *alignment* task (Fig. 6.2).

Through a series of real-world experiments, studying a range of tasks such as grasping, pouring, and inserting objects, we show not only that DINOBot achieves one-shot imitation learning on tasks where existing methods require many demonstrations, but it also achieves very efficient generalisation to novel objects. Our conclusion, and the key takeaway message, is clear: **designing an imitation learning framework around image-level (retrieval) and pixel-level (alignment) visual tasks allows us to leverage the remarkable capabilities of DINO-ViTs, leading to unprecedented learning efficiency compared to alternative paradigms**.

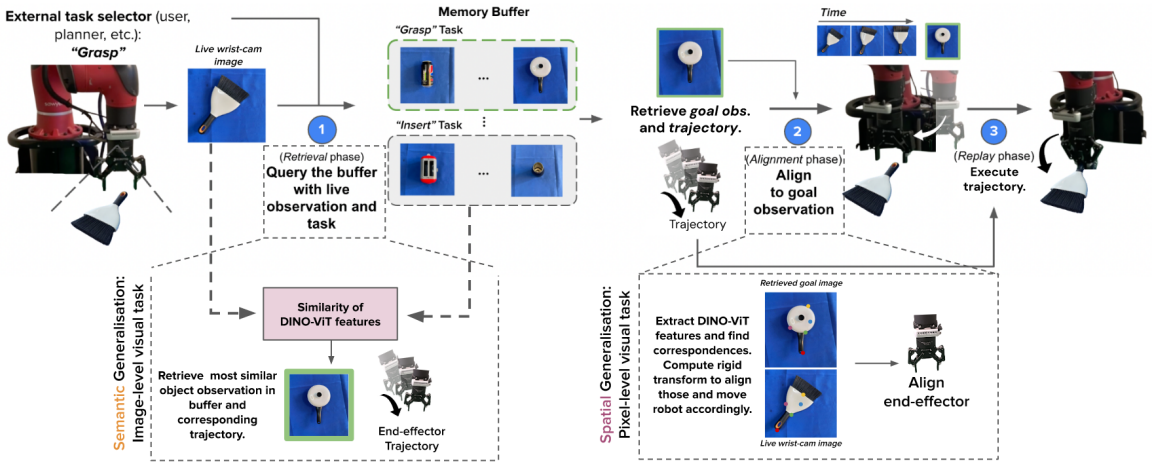


Figure 6.2: Overall illustration of our framework. Upon observing a new object, the robot visually compares it other objects observed during demonstrations to find the most similar object (*semantic, image-level reasoning*), and retrieve both its image and the trajectory executed on that object. Then, the robot aligns its end-effector with this image (*spatial, pixel-level reasoning*), before then executing that trajectory. These two phases of reasoning are both based on extracting and matching DINO-ViT features.

6.2 Related Work

Imitation Learning on top of Foundation Models. Many recent imitation learning methods for robot manipulation train end-to-end policies, built upon features extracted from pre-trained models. However, such methods still require tens or hundreds of demonstrations per task [179, 147, 164]. In our work, we show that an explicit decomposition into an image-level retrieval phase and a pixel-level alignment phase followed by a trajectory replay, results in a substantial improvement in data and time efficiency.

Retrieval for few-shot learning. Related to our use of DINO's image-level capabilities, retrieval for few-shot learning, especially in robotics, reinforcement learning and control, has also been investigated elsewhere [129, 23, 8, 181, 178, 137, 155, 53, 105]. Some approaches directly retrieve the actions to execute [137, 155, 181] or the data to train a policy [53]. [105] also retrieves the most similar demonstration from a buffer, and then aligns the robot by using optical flow between the current and goal observations, but is considerably worse at generalising to new objects as we demonstrate in Sec. 6.4.

Local correspondences for robot manipulation. Related to our use of DINO's pixel-level capabilities, extraction of dense or sparse correspondences through neural networks has been explored elsewhere for robot manipulation [68, 138, 195]. However, these methods require object-specific data collection to train networks. [196] uses a general keypoint-tracking network, that however does not generalise to unseen objects as we will illustrate in later sections. [79, 83, 10] demonstrate the capabilities of off-the-shelf DINO-ViT features for pose estimation. On

top of these, we build a full IL framework based on retrieval and alignment. [200] trains a graph neural networks that takes as input local geometry embeddings of point clouds to perform few-shot IL. [207, 111, 206] also use DINO features to find matches and keypoints, they however mostly focus on grasping, while DINOBot can learn a large repertoire of everyday skills.

Trajectory decomposition for robot manipulation. Our work is related to [110, 192, 198, 154], which also decompose robot manipulation into visual alignment and then replay of a trajectory. [110] trains an object-specific visual servoing policy with autonomously collected data, while [192] trains a general goal-conditioned alignment policy in simulation. However, neither of these methods enable generalisation to novel objects, and furthermore, with DINOBot we demonstrate that, by using DINO-ViTs, neither object-specific data collection nor additional simulation training is needed.

Semantic and spatial reasoning. CLIPort [182], like DINOBot, also phrases object manipulation as semantic reasoning combined with spatial reasoning. However, there are some fundamental differences: 1) their pipeline is more implicit, going from language and visual observations to affordance prediction through a single forward pass of a two-streams network, while we have explicit retrieval and alignment phases, that leads to DINOBot's better efficiency 2) their method is designed for top-down pick-and-place-like tasks, while we experiment with more complex tasks, also in a 6-DoF environment like a kitchen.

6.3 Method

Figure 6.2 illustrates our framework. During deployment, DINOBot executes manipulation tasks via a semantic *retrieval* and a spatial *alignment* phase, followed by a demonstration *replay* phase. During training, the operator provides demonstrations, and the collected data fills a memory buffer. For each demonstration, this data consists of a wrist-camera image captured at the beginning of the demonstration, and the end-effector trajectories for the demonstration. During testing, for the *retrieval* phase, the robot queries the memory buffer with its current wrist-camera observation. Given the nature of the task to be executed (e.g. "Grasp" or "Insert"), it retrieves the most similar image which is then used as goal image (Fig. 6.3), and the corresponding end-effector trajectory. The interaction with the object is decomposed into the *alignment* and *replay* phases. It first uses the retrieved goal image to align the end-effector with the test object correctly. Then the robot replays the recorded demonstration trajectory. Our method only needs an RGB-D wrist camera, rigidly mounted to the robot's end-effector. No prior knowledge of objects or tasks is needed, and no external camera is needed.



Figure 6.3: In each column, given a live image (top) DINOBot retrieves from the buffer the most similar image in the buffer (bottom), and finds correspondences between the two.

Spatial Generalisation through Alignment and Replay

To teach the robot how to interact with an object, the human operator controls the robot to provide a demonstration. The end-effector starts from the *bottleneck pose*, B^O, which is a pose arbitrarily chosen by the operator to start the demonstration from, and from where the object must be visible from the wrist camera. The demonstration is recorded as a sequence of 3D linear and 3D angular velocities of the end-effector, expressed in the end-effector frame E. This sequence is the trajectory s, such that $s = [\mathcal{V}_1^E, \ldots, \mathcal{V}_T^E]$, where $\mathcal{V} = [v_x, v_y, v_z, w_x, w_y, w_z]$. When starting the demonstration from pose B^O, the robot also records the wrist-camera observation of the object from that pose, which we call the *bottleneck observation*. For each demonstration, a *task* name is also specified by the operator, e.g. "Grasp", "Insert", or "Pour". In summary, each time a new demonstration is recorded by the operator, the framework adds the following data to the memory buffer: the bottleneck observation recorded before starting the demonstration, the trajectory of velocities $s = [\mathcal{V}_1^E, \ldots, \mathcal{V}_T^E]$, and the task name. These concepts are analogous to the ones described in the previous chapters, with the addition of short language annotations for the tasks.

We can imagine the bottleneck pose being rigidly attached to the virtual (since we do not assume object models) object frame. With W being the world frame, when we move the object, the global bottleneck pose B^W moves rigidly with the object, whilst the local bottleneck pose B^O stays constant. If the end-effector is re-aligned to B^O when the object is moved, replaying the end-effector velocities of the recorded trajectory would suffice in solving the task [110]. Thus, the robot only needs to reach B^O again for novel poses of this object, which is now expressed as a different B^W.

When a new object is observed during testing, how can the robot align itself with the object to reach B^W again? Given the observation recorded at the beginning of some demonstration stored in the buffer, and the current live observation recorded from the wrist-camera, DINOBot performs visual servoing to align the two images. Specifically, we extract deep patch features [51] from the DINO-ViT and use the method described in [10] to find correspondences through a Best Buddies Nearest Neighbour matching phase 6.3). This procedure generates two lists of keypoints, defined as 3D coordinates $C_1 = \{x_{1,i}, y_{1,i}, z_{1,i}\}$, $C_2 = \{x_{2,i}, y_{2,i}, z_{2,i}\}$ (we extract the depth of each RGB pixel correspondence through the RGB-D wrist camera). We then compute the least-squares rigid transformation that aligns the two lists of corresponding keypoints [185, 150] and move the robot accordingly. We repeat this process until the alignment is precise enough, i.e. the norm of the distance of the two lists of correspondences is smaller than a threshold. Once the alignment is completed, replaying the recorded trajectory allows the robot to successfully interact again with the object. We provide additional info and code on the website.

Semantic Generalisation through Retrieval

In the previous section, we described how we formulate the interaction with an object as visual alignment with a goal image, followed by replay of a trajectory. But given a new object to interact with, how does the robot select the best goal image and trajectory from its memory buffer? First, we assume that the robot receives, either by the human operator or by an external planner, the *task* to execute with the object, e.g. "Grasp" or "Open". The robot then records a live observation of the object from its wrist-camera. It then *retrieves* from its memory buffer the *bottleneck observation* most similar to the live observation, from the subset of the memory buffer for that task (e.g. if the task is to "Grasp", then all demonstrations which were also "Grasp" tasks would be considered). DINOBot performs retrieval by extracting features for each observation in the buffer and the live observation through the DINO-ViT. In our case, we extract the CLS token, a 1×768 vector [51]. It then performs a nearest neighbour search by computing the cosine similarity between the extracted features, as described in [152], to find the closest match in the buffer. The best match is retrieved together with its recorded trajectory: these are then used as goal observation for the *alignment* phase and actions to be replayed during the *replay* phase. As such, DINOBot is able to generalise observed demonstrations to novel objects, by independently using both the image-level and pixel-level capabilities of vision foundation models.

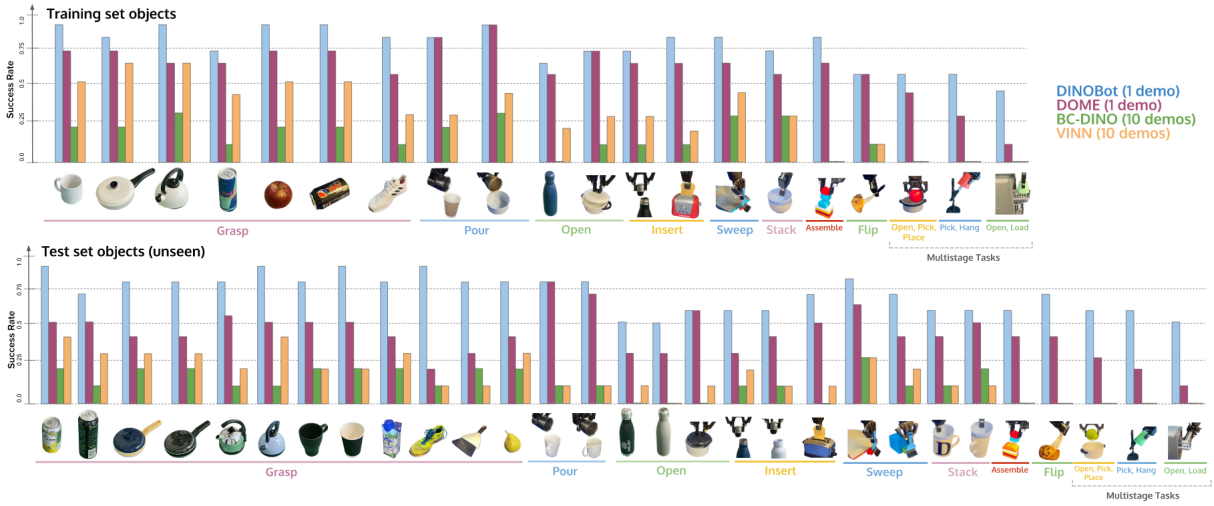


Figure 6.4: Success rates on each object for all methods, tested with 10 episodes per object. Here we also report the average success rate with 95% confidence Wilson intervals:
Training set objects: DINOBot: 0.74 [0.68, 0.80], **DOME**: 0.61 [0.54, 0.67], **BC-DINO**: 0.14 [0.10, 0.19], **VINN**: 0.14 [0.10, 0.20].
Test set objects: **DINOBot**: 0.71 [0.66, 0.76], **DOME**: 0.43 [0.37, 0.49], **BC-DINO**: 0.11 [0.08, 0.15], **VINN**: 0.16 [0.12, 0.21]

6.4 Experiments

In this section, we empirically measure the ability of our method to learn behaviours efficiently and transfer them effectively to new objects, and we structure our experiments around several important questions which we present in the following pages. These experiments were conducted on two different real-world environments, on a total of **15 tasks** with **53 objects** (a task, in this work, can be intuitively described as a verb, like *grasp* or *open*, and be applied to many different objects). First, a tabletop environment with 49 everyday objects (20 train, 29 test) and 8 types of tasks: **grasping**, **pouring**, **opening**, **inserting**, **sweeping**, **stacking**, **assembling**, **flipping** and 3 *multi-stage tasks* following the demonstration procedure from the previous chapters: **open+pick+place**, **hanging cups** and **opening and loading a dishwasher**. Second, a toy kitchen environment where we teach 4 tasks: **opening** the microwave, **inserting** a plate into a dishwasher, **placing** a plate in the sink, and **turning** an oven knob. To prove the effectiveness and generality of DINOBot, we took these tasks from other recent imitation learning papers, and we therefore test on *all the tasks* from: 1) **Relational-NDF** [183], (where instead of *bottle in container* we have multiple, more precise insertion tasks) + 2) **FISH** [85], (non multifingered ones - as *door opening*, we have *open a microwave door* - for *key insertion*, we have several precise insertion tasks with a lower than 5mm error tolerance), + 3) **VINN** [155], (where instead of *pushing* directly, we sweep, i.e. push with a tool), + 4)

Relay Policy Learning [82]. We invite you to watch the videos on our website at robot-learning.uk/dinobot.

Setup: We run our experiments on a Sawyer robot with 7 degrees of freedom mounting a Robotiq 2F-85 gripper. The robot is controlled via position control during the alignment phase and velocity control during the interaction phase. We use a single wrist-mounted Intel RealSense D435 camera. The camera receives RGB-D images which we rescale to $224 \times 224 \times 4$. The robot starts learning *tabula rasa*: no previous knowledge of tasks or objects, such as CAD models, is used.

Baselines: We compare DINOBot to the following baselines. 1) **DOME** [192], a similar one-shot IL method based on a learnt (through simulation), goal-conditioned visual servoing phase followed by a trajectory replay. However, DOME does not generalise to novel objects, and so we extended the original method with our retrieval phase to provide the goal image to align with. This baseline exists to compare DINO's pixel-level alignment abilities with DOME's simulation-trained visual servoing network. 2) **BC-DINO**, a behaviour cloning (BC) implementation that trains a network to output actions on top of features extracted through a DINO-ViT. This baseline exists to compare our framework to the more typical approach in recent works, where BC policies are trained upon features extracted from pre-trained models [147, 179, 164] on all the collected demonstrations. 3) **VINN**: Visual Imitation through Nearest Neighbours (VINN) [155], frames learning from demonstration entirely as a retrieval problem, where actions are computed by retrieving the k most similar demo observations and averaging their corresponding actions. We use DINO features to embed the observations and retrieve them.

All methods receive as input the task to execute and the wrist-camera observation. For DINOBot and DOME, we provide a single demonstration for each training object. As BC-DINO and VINN are not designed as one-shot IL methods, we provide 10 demonstrations per training object. More details on the implementation of these methods and on the experiments can be found on our website.

Can DINOBot learn everyday-like tasks efficiently, and transfer those skills to novel objects? How does it compare to baselines from recent literature? To answer this, we train each method with demonstrations of how to interact with the objects in the training set on the tabletop environment (Fig. 6.4, top). We then test each method using the objects depicted in the figure: both the training objects (top), and the unseen test objects (bottom).

At test time, we position each object randomly on the table, and inform the robot what the task to perform is (e.g. "Grasp", "Open", etc). We sample a position inside a 40cm $\times$ 40cm area, and an angle between -45° and 45° relative to the original demonstration angle. We run 10 trials per object, sampling a new object pose each time. In this tabletop scenario, the alignment phase is 4-DoF, with the robot always aligned with the vertical axis and only rotating around this axis, while the *replay phase* trajectory execution is 6-DoF. Later in the chapter, we will also explore 6-DoF alignment in the kitchen environment.

The performance of each method on each object is shown in Fig. 6.4. We group the results into training set objects (for which the robot received demonstrations, to study one-shot IL), and test set objects (to study generalisation to novel objects).

The results show that not only can DINOBot obtain remarkable one-shot performance on the training objects, but that it can generalise to unseen objects: performance stays considerably close to the training set performance.The performance of the baselines is noticeably lower, illustrating the benefits of our combination of retrieval and alignment to leverage the capabilities of DINO-ViTs. Using the extracted features as input to train a BC network, perhaps the most typical paradigm for using pre-trained networks in IL [147, 164, 179], clearly performs worse, even with 10 times the number of demonstrations. DOME performs well on training set objects, but performance degrades when trying to generalise to unseen objects, as the method was not designed to do so. Furthermore, the large amount of episode we ran to evaluate the performance of each method allows us to draw strong statistical guarantees regarding the superior performance of our proposed method, as there is no overlap over the 95% confidende Wilson intervals listed in 6.4.

Can DINOBot learn to interact with a complex kitchen environment? In this section, we now study more complex 6-DoF tasks and environments. We provide a single demonstration to our robot for the following tasks: 1) open a microwave, 2) insert a plate into the dishwasher, 3) put a plate into the sink, and 4) turn an oven knob. After each demonstration, we test the ability of our method to replicate the task with the robot starting from a different starting position, with 10 trials for each task. As the only input the robot receives are wrist-camera observations and no proprioceptive data, moving the robot to a different initial state is akin to moving the kitchen to test for spatial generalisation. In this experiment, we skip the *retrieval* phase to study the *alignment* and *replay* phases' one-shot IL ability in isolation, such that the robot is asked to perform the same task it has just been shown from the demonstration. The robot now has to perform a challenging 6-DoF alignment by extracting correspondences

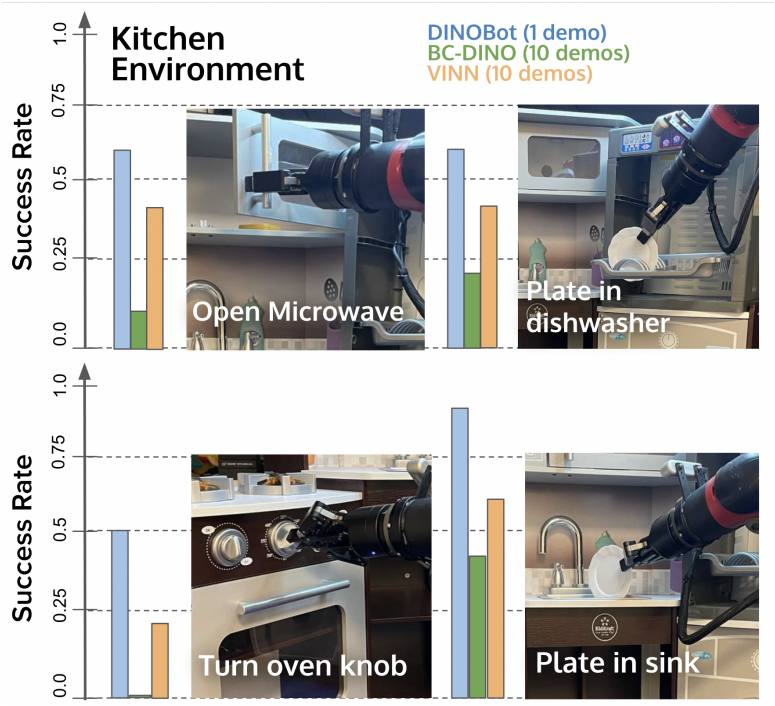


Figure 6.5: Success rates of our method and several baselines on complex kitchen tasks. Here we also report the average rate with 95% confidence Wilson intervals.
DINOBot: 0.65 [0.50, 0.78], **BC-DINO**: 0.17 [0.09, 0.32], **VINN**: 0.40 [0.26, 0.55]

between the *bottleneck observation*, recorded at the start of the demonstration, and its live wrist-camera observation (videos of execution and keypoints extractions on our website). We compare DINOBot with one demonstration to **BC-DINO** and **VINN** with 10 demonstrations. DOME is designed for 4-DoF tabletop settings and cannot be deployed for this task.

Results shown in Fig. 6.5 clearly show that our framework can successfully solve these everyday-like tasks, when receiving only a single demonstration. These results further demonstrate how the use of DINO extracted features to find visual keypoints can lead to superior performance than using those features as inputs to a policy network.

Does DINOBot work even in the presence of distractor objects? In this experiment, we provide a single demonstration for each of the following tasks: 1) **grasping** a can, 2) **inserting** bread into a toaster, and 3) **pouring** from a cup into a mug. Here we study the *alignment* and

Method	w/out Distractors	w Distractors
DINOBot (1 demo)	**0.8** [0.63, 0.90]	**0.76** [0.58, 0.88]
DOME (1 demo)	0.67 [0.49, 0.81]	0.5 [0.33, 0.67]
BC-DINO (10 demos)	0.2 [0.10, 0.37]	0.07 [0.02, 0.22]
VINN (10 demos)	0.33 [0.19, 0.51]	0.07 [0.19, 0.51]

Table 6.1: Success rates with and without distractors, with 95 % confidence Wilson intervals.

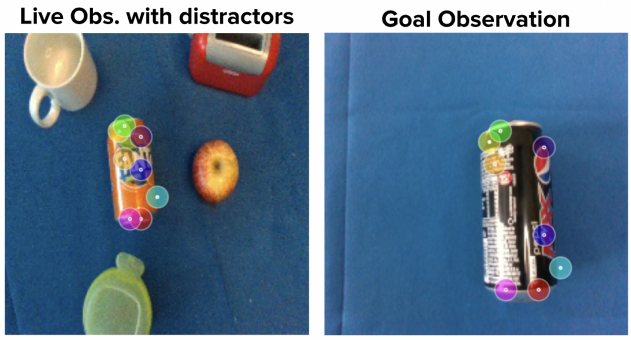


Figure 6.6: Despite the distractors, sensible correspondences can still be found.

replay phases robustness to distractors in isolation, hence we provide the goal image manually, skipping the *retrieval* phase. At test time, we use an unseen test set object (e.g. a different can or a different toaster), but we also vary the scene from the demonstration setup by adding a set of distractors from different classes (Fig. 6.6). We compare DINOBot's performance against the baselines: DOME receives a single demonstration as well, while BC-DINO and VINN receive 10 demonstrations.

We run 10 test trials for each task, with and without distractors. Results in Table 6.1 demonstrate that DINOBot not only surpasses all the baselines, but faces the smallest decrease in performance due to distractors. This demonstrates that DINO-ViT is very robust and able to extract correct correspondences between the goal observation object and the live observation, even in the presence of additional distractor objects(Fig. 6.6).

What technique performs best in retrieving the most similar observation from the memory buffer? Being an essential part of our method (Sec. 6.3), we investigate the performance of different retrieval techniques to further motivate our design choice.

We compare retrieval based on features extracted from publicly-available ImageNet-trained ResNet-50 [89], CLIP [163], R3M [147] and DINO-ViT [31], each of which has been pre-trained on vast datasets of images (refer to the website for additional details). We measure retrieval accuracy by providing to each method an observation of an unseen object from the test set, composed of the 20 objects of Fig. 6.4 (bottom). We then measure how many times the method retrieves from the buffer an observation of the object belonging to the same class. We show results in Figure 6.7. The features extracted via a DINO-ViT achieve the best accuracy, confirming recent findings from the literature [31, 10] that such features effectively encode semantic, geometric, global and local information of the observation.

How accurate is the keypoints-based alignment phase of DINOBot? Here, we evaluate alignment precision in isolation, comparing it to two other alignment techniques from recent literature: **RoboTAP** [196], which uses keypoints extracted through **TAPIR** [49] for visual servoing and alignment, and **FlowControl** [13, 105], which uses optical flow to perform alignment; for the optical flow network, we use **RAFT** [190]. We use three pairs of objects: shoes, cans, and toasters. We generate a series of top-down observations of the objects where we manually translate and rotate the objects, in addition to artificially changing the background. Each method is then tested on its accuracy to compute the correct translation and rotation given an initial observation and a transformed observation. We test each method both when the pair of images depict the same object, and when the two images contain two different objects of the same class, e.g. two different shoes, to evaluate for semantic and geometric robustness.

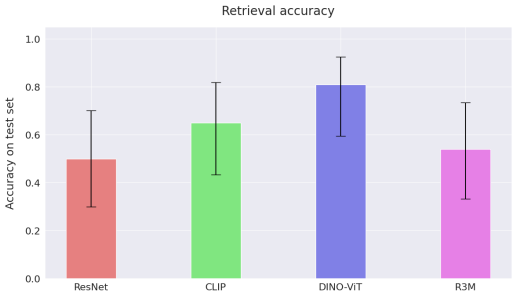


Figure 6.7: Comparison of retrieval accuracy of the baselines, with 95% confidence Wilson intervals.

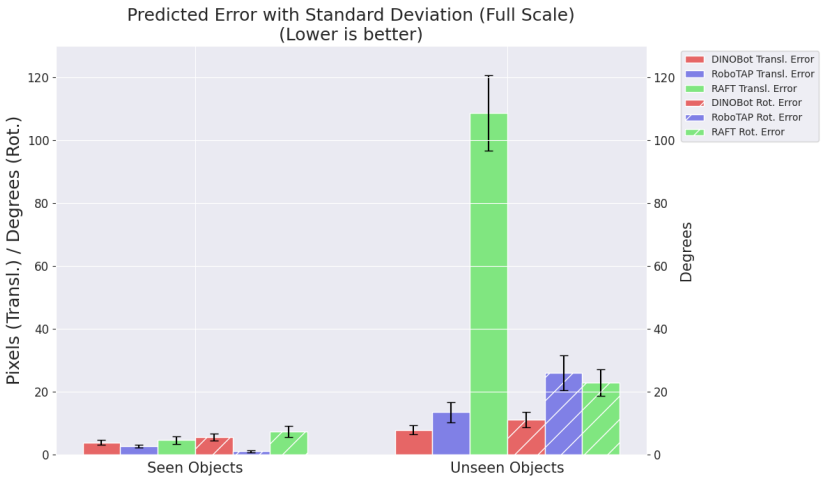


Figure 6.8: Alignment accuracy benchmark: we measure average translation (in pixels) and rotation (in degrees) error at finding the correct translation and rotation between a goal and a current observation. We also show 1 standard deviation interval.

Results in Fig. 6.8 demonstrate that, while RoboTAP is extremely accurate when the same object appears in the pair of observations, performance degrades considerably when matching two different objects, albeit belonging to the same class. RAFT shows a similar trend, although with overall worse performance. DINOBot instead is able to generalise to unseen objects, maintaining strong performance.

How much can DINOBot generalise to objects with different sizes and appearance from a single demo? Here, we collected a new set of

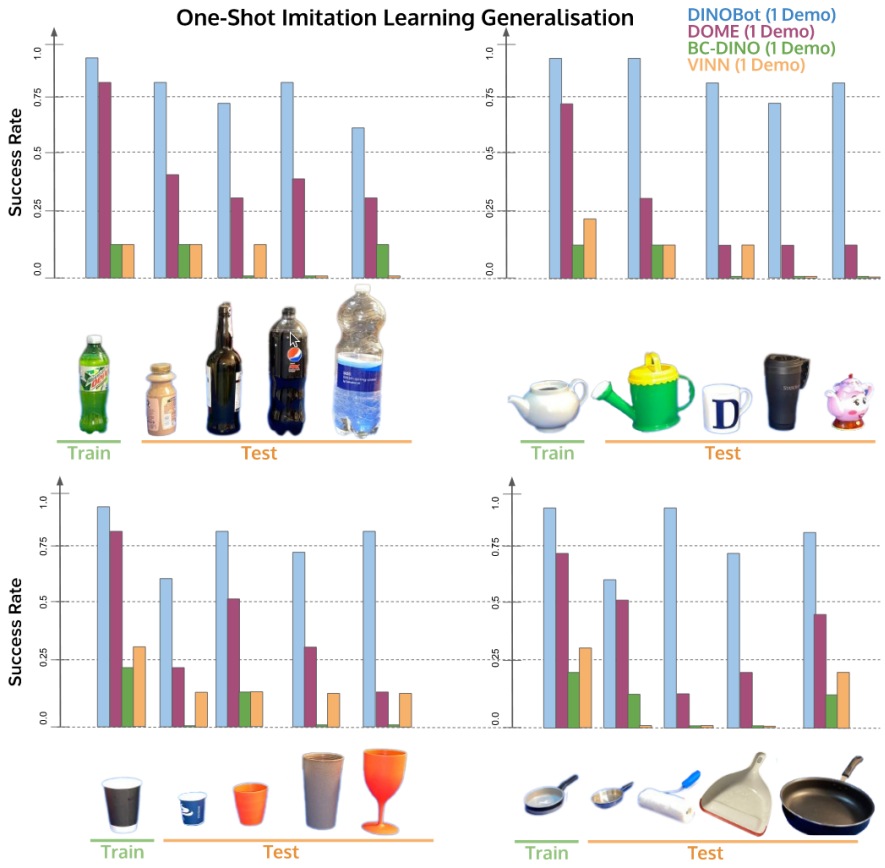


Figure 6.9: Success rates for the one-shot, intra-class IL generalisation experiments, where we provide a single demo on an object and test generalisation both to the same and to unseen objects of the same class.
We also provide average success rates aggregated over all 4 objects classes, with 95% confidence Wilson intervals.
DINOBot 0.78 [0.72, 0.83], **DOME** 0.36 [0.30, 0.43], **BC-DINO** 0.06 [0.03, 0.10], **VINN** 0.10 [0.06, 0.14]

objects belonging to 4 classes - bottles, kettles, cups, and pans - where the object sizes and shapes vary significantly within each class. For each class, we then provide a demo for a single object, and then test on all the objects - 1 demo object, and 4 unseen objects - measuring the success rate over 10 test trials for each. As such, here we investigate the ability to generalise to novel objects that can vary substantially in size and shape, e.g. bottles more than double in size, mugs with handles of different shapes and position, etc. Results in Figure 6.9 show that DINOBot achieves significant generalisation and again outperforms baselines.

How does DINOBot compare to other methods across three challenges: adaptability, dexterity, and precision? To better highlight the different manipulation abilities expressed by our method, we cluster the results for the tasks in 6.4 into three groups, which cover some of the main abilities a robot should possess to tackle everyday-like tasks: **adapt-**

ability, **dexterity**, and **precision**.

We cluster as follows. 1) All results from *grasp* experiments are grouped into **adaptability**, as grasping is the task that encounters the largest variety in objects [63], and thus requires significant adaptation to novel shapes based on visual observation. 2) All results from *pour, sweep, flip, open dishwasher* are grouped into **dexterity**, as these require non-trivial, often fast movements to succeed. 3) All results from *insert, assemble* are grouped into **precision**, as these have lower than 5mm positional tolerance. Results, illustrated in Fig. 6.10, demonstrate that DINOBot surpasses all baselines on all these challenges. We observed that the use of retrieval helped DINOBot adapt to unseen objects of various shape, while alignment and trajectory replay helped tackling precise and dexterous tasks, strongly surpassing the end-to-end baselines.

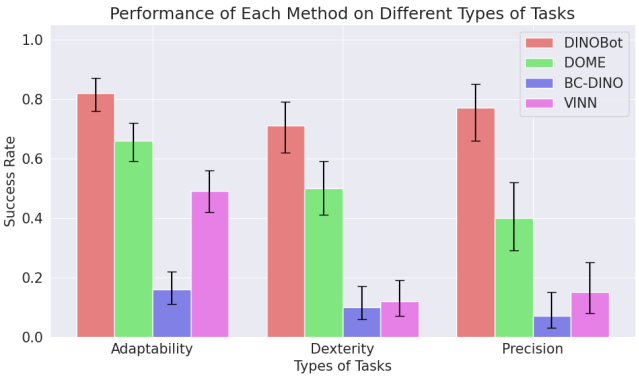


Figure 6.10: Performance of each method on tasks that require adaptability, dexterity or precision. We show average success rate and 95% confidence Wilson intervals.

6.5 Limitations

The current method relies on the effectiveness of vision models, both in the retrieval and alignment phases. While we empirically see how the performance of this model has been growing substantially, we still rely on off-the-shelf models. Finetuning on robotics data might further improve performance. Like the methods in the previous chapters, the method is not suited for tasks that rely on the relative position of objects between each other, or tasks that rely on dexterous, closed-loop interaction with objects.

6.6 Discussion

We have introduced DINOBot, an imitation learning framework designed around image-level (retrieval) and pixel-level (alignment) visual tasks, to take full advantage of the abilities of DINO-ViT foundation models. Our extensive experimental investigation demonstrated how this framework surpasses other DINO-based baselines, that do not harness the full capabilities

of these vision models.

We demonstrated how we can leverage the abilities of vision models to design more efficient learning algorithms that can efficiently tackle a certain class of problems. In the next chapter, we will continue this investigation on the use of large, pretrained models. We will see, however, that even models trained entirely on textual data, a modality far from robotics data, can become effective backbones to efficiently teach skills to robots.

Chapter 7

Keypoint Action Tokens Enable In-Context Imitation Learning in Robotics

7.1 Introduction

A robot's ability to learn new skills from just a few demonstrations, known as few-shot imitation learning, is important for scalable robot learning [55]. However, learning robust skills that can generalise to conditions not seen during the demonstrations is very challenging [39]. Whilst recent successes in imitation learning have shown that scaling up behavioural cloning can sometimes achieve such generalisation [56, 39], this requires collecting huge datasets of demonstrations covering diverse objects and conditions, and we are still lacking effective methods for

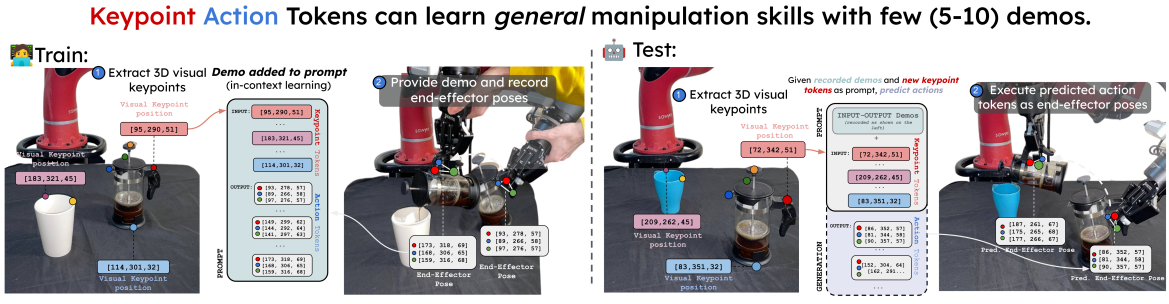


Figure 7.1: We introduce Keypoint Action Tokens (KAT), a method to repurpose off-the-shelf, large text-pretrained Transformers for efficient imitation learning, by representing sequences of observations and actions as sequences of keypoint-based tokens, converted into text and added to a prompt. KAT can learn general behaviours with as few as 10 demonstrations, and since no training is needed, learnt skills can be deployed immediately after the demonstrations.

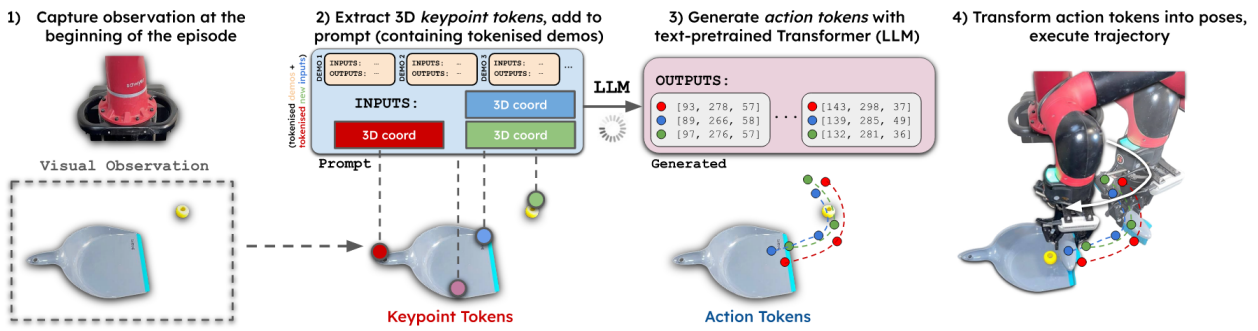


Figure 7.2: An illustration of our pipeline. KAT transforms a visual observation into a *sequence of keypoint tokens*. From it, the text-pretrained Transformer (LLM) predicts *action tokens*, that are then executed as a trajectory of poses.

learning of entirely novel skills from just a few demonstrations.

Meanwhile, recent works have demonstrated how Transformers [193] excel in the ability to learn sequence-to-sequence patterns from few examples [29, 149]. Interestingly, a remarkable recent observation is that Transformers can effectively learn to emulate patterns even when pre-trained on data from vastly different domains, acting therefore as *general pattern machines*. In particular, large Transformers pre-trained on text have demonstrated the ability to quickly learn sequence-to-sequence patterns that go well beyond text-based reasoning tasks [142]. Building upon this observation, in this chapter we now investigate the following: can we repurpose off-the-shelf large Transformers, pre-trained on language where data is abundant, to act as *few-shot imitation learning machines* in robotics, where data is scarce?

We study this by modelling imitation learning as a sequence-to-sequence prediction problem, with the input sequence being observations and actions from the demonstrations, plus the robot's current observation, and the output being the sequence of actions the robot should execute. However, a key challenge here is grounding text-based Transformers in vision, such that these sequences, which represent visual observations and end-effector actions, can be expressed as language tokens. We therefore introduce **Keypoint Action Tokens (KAT)** 7.1, a framework that makes use of off-the-shelf Vision Transformers [50] and text-based large Transformers [29] to create a *low-level language of robotics actions*. By converting the sequences of observations and actions into a *sequence of tokens*, we show that a large Transformer, whilst pre-trained only on text, can learn to emulate complex physical behaviours from only a few example demonstrations (≤ 10).

We investigate Keypoint Action Tokens through a series of real-world experiments, providing insights into performance across a variety of few-shot learning challenges, such as spatial generalisation, generalisation to novel objects, learning multi-modal behaviours, and learning

to perform trajectories with 6-DoF action spaces. We show that with Keypoint Action Tokens, off-the-shelf text-pretrained Transformers are able to perform few-shot learning on everyday tasks and are on par or superior to the current state-of-the-art in imitation learning (diffusion policies [35]). To the best of our knowledge, this is the first work to propose a comprehensive method to perform imitation learning from vision using an off-the-shelf large text-pretrained Transformer as the core imitation learning algorithm, with no additional training. A key contribution of our work is strong evidence that the **progressive evolution of capabilities of large pretrained Transformers is leading to the emergence of more and more general and efficient pattern learning machines, that can directly be used off-the-shelf to tackle sequence-to-sequence imitation learning *without the need for training on any robotics data***. The ability to repurpose large networks trained on language domains, where data is abundant, is a promising avenue to unlock unprecedented learning efficiency in robotics, where data is scarce. Videos are available on our website https://www.robot-learning.uk/keypoint-action-tokens.

7.2 Related Work

Few-shot Imitation Learning. Few-shot imitation learning is one of the main desiderata in robot learning, and more generally, in machine learning. Behaviour cloning, where a neural network is trained over a dataset of observation-action pairs coming from an expert demonstrator, often requires hundreds of laborious demonstrations to learn tasks. Over the years, many methods have been proposed to tackle this problem. [55] proposed to train a network to explicitly take in a demonstration as input, and output actions that would follow the behaviour demonstrated. [107] proposed to learn a Siamese network on pairs of demonstrations of the same behaviour to learn task-specific embeddings for one- or few-shot learning. [38] proposed to train several imperfect dynamics model to train a policy to perform efficient adaptation to real-world data. All these methods, however, still rely on gathering a substantial amount of interaction data. We propose to use the emergent pattern completion abilities of large Transformers pretrained solely on text, where no robotics data is used to pretrain or finetune it.

Finetuning Large Pretrained Models An effective paradigm that emerged from the recent research is to first pretrain large models on enormous, task-agnostic datasets, and then perform finetuning with substantially smaller datasets representing the task at hand. Such models are referred to as Foundation Models [62]. While being extremely successful in the fields of computer vision and natural language processing, the lack of vast robotics datasets hindered

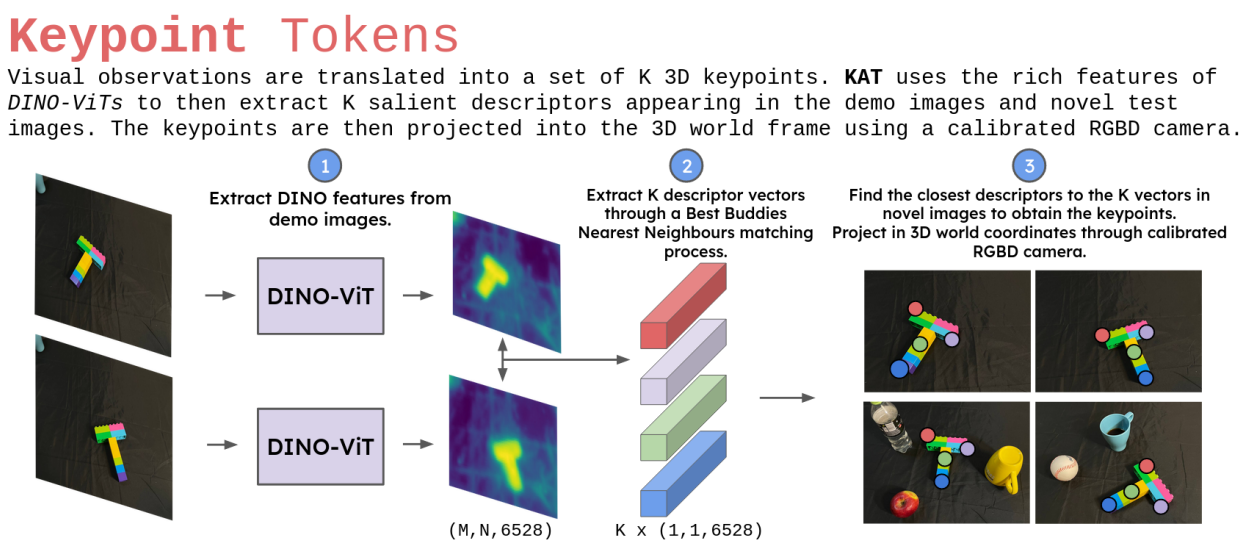


Figure 7.3: Illustration of the extraction pipeline of *keypoint tokens*.

the creation of robotics specific Foundation Models. While [56, 39] demonstrated promising results by finetuning large Vision-Language Models on robotics data, learning new tasks is still inefficient and laborious. With Keypoint Action Tokens, we show that no fine-tuning is needed and thus we can achieve highly efficient in-context learning.

Use of Large Language Models in robotics as planners. The exponential progress in capabilities of text-pretrained Transformers, or Large Language Models (LLMs), to interpret and reason over text generated a plethora of robotics works leveraging such models. [60, 126, 197, 97, 52] use LLMs as high-level planners, obtaining a list of steps to solve tasks in language form. [98, 214] leverage LLMs to map a high-level description of a task in natural language into rewards that can be used by a trajectory optimiser. While effective, these methods rely on LLMs to generate high-level plans or guidance for external optimisers, and therefore need additional techniques to ground these outputs into actions. In our work, we *do not use LLMs for natural language*, but instead exploit their pattern learning capabilities to directly process sequences of observations and generate executable low-level trajectories. We therefore shed light on a counter-intuitive phenomenon: ***one of the most effective uses of LLMs in robot learning is achieved without the use of natural language, neither in the inputs nor in the outputs.***

Language Models that output robotic actions. [56] finetunes a Vision-Language Model with robotics data in the form of language-annotated demonstrations. They therefore obtain a model that can receive as input a language instruction, and can output, similarly to our method, actions that can directly be executed by the robot. The main differences, however are that 1) we do not need finetuning, instead employing off-the-shelf models, and 2)

while their model receives language instructions, our method can perform *in-context learning* on low-level demonstrations, therefore acting as a general imitation learning machine. [118] demonstrated the ability of LLMs to directly output trajectories of executable actions instead of high-level textual plans. However, this can only generate and execute trajectories based on language instructions and thus being unable to learn from demonstrations. [142] was one of the first works to show the general pattern learning abilities of LLMs. They demonstrated the capability of LLMs to autoregressively complete a partial trajectory received as an input, therefore learning patterns beyond human language. However, in contrast to their work, we propose a full imitation learning framework that receive complex visual inputs and replicate expert behavior on many everyday tasks, that present a set of unique challenges we will later discuss. To the best of our knowledge, we are the first work to propose a complete framework that can learn from demonstrations in the forms of sequences of visual observations and actions using off-the-shelf text-pretrained Transformers.

In-Context Learning Scaling the training of large Transformers, especially on language data, led to the emergence of in-context learning [29]. While the explanation for the phenomenon is not entirely clear, recent works have investigated its emergence in simple settings [149]. [199] investigated the parallels between in-context learning, a purely feed-forward adaptation mechanism with no weights updates, and gradient descent, hypothesising that Transformers learn to implicitly apply a gradient update via attention heads. These observations inspired this work, and the use of Transformers and their in-context learning ability as general imitation learning machines. Note that Transformers are not the only way to obtain in-context learning. [201] trained a graph neural network to learn object alignments when conditioned on a few demonstrations in an in-context learning manner. However, they represent imitation learning as learning optimal objects alignments, while we represent policies exclusively as a sequence of robot poses, proposing both a new input-output representation (keypoint and action tokens) and the use of pretrained Language Models as in-context imitation learning machines.

7.3 Method

We formulate few-shot imitation learning as a sequence-to-sequence problem. Let $\mathcal{O}$ represent the sequence of observations received by the robot and $\mathcal{A}$ the sequence of end-effector actions - in our case a trajectory of $SE(3)$ poses $T^{\mathcal{W}}$ expressed in the world frame $\mathcal{W}$. Let $\mathcal{D} = [d_1, \ldots, d_n]$ be a sequence of demonstrations given by a human expert, with each $d_i = [\mathcal{O}_i, \mathcal{A}_i]$ representing the desired sequence of actions given a certain sequence of observations. Our goal is to find

an effective *imitation learning machine* $\mathcal{M}(\mathcal{D}, \mathcal{O}_t) \rightarrow \mathcal{A}_t$, with $\mathcal{O}_t$ being a novel sequence of inputs and $\mathcal{A}_t$ being a predicted sequence of actions that would effectively emulate the expert behaviour implicitly represented in $\mathcal{D}$.

In this work, the robot records a visual observation, o_i, at the beginning of the episode. Therefore, for each demonstration d_i, the robot collects an observation o_i and a sequence of actions $a_{i,1:T}$. When deployed, the robot's task is to record a new observation o_j and compute a sequence of actions $a_{j,1:T}$ that emulate the expert behaviour observed in $\mathcal{D}$. The pipeline is illustrated in Fig. 7.2. Recording and using a single visual observation is a design choice of this work and not an inherent limitation of our method, which could receive a new visual observations every T steps, and compute a new trajectory of actions conditioned each time.

The current literature often decomposes the problem in two steps: first, the output of $\mathcal{M}(\mathcal{D})$ is a policy network π_θ trained on $\mathcal{D}$; second, the sequence of actions is computed as $\pi_\theta(\mathcal{O}_t) \rightarrow \mathcal{A}_t$. We demonstrate, however, that this decomposition is not necessary: we can instead leverage the *in-context learning* abilities of pretrained Transformers to input both $\mathcal{D}$ and $\mathcal{O}_t$ to the network, and then compute $\mathcal{A}_t$ emulating the patterns observed in $\mathcal{D}$.

More specifically, *in-context learning* [149, 29, 199] commonly refers to the ability of certain networks to perform few-shot learning *without* the need to update their parameters. The network is conditioned on both a few example datapoints, and the new input from which it needs to compute an output following the behaviour expressed in the examples, taking the form $f_\phi(\mathcal{O}_t, \mathcal{D}) \rightarrow \mathcal{A}_t$. While not entirely exclusive to them [201], the recent literature suggests that this ability emerges from large Transformers trained to perform autoregressive pattern completion on vast datasets, the most successful case being language [149, 29].

The core finding of this work is that pretraining or finetuning using robotics data *is not needed*, and we can instead **repurpose a large Transformer pretrained entirely on textual data**, (a Large Language Model), without any additional finetuning, and it can act as an extremely efficient *imitation learning machine* $\mathcal{M}$.

To unlock these abilities, we first need to transform $\mathcal{O}$ and $\mathcal{A}$ into a data format that can be interpreted by text-pretrained Transformers, that ingest *text*. In this work, *tokenising* refers to transforming observations and actions into a form that can be given as input to a text-pretrained Transformer/Large Language Model, as we will shortly describe in more detail.

Our goal is to represent $\mathcal{O}$ and $\mathcal{A}$ as a sequence of tokens, finding a *language* that represents observations and actions effectively. To do so, we introduce **Keypoint Action Tokens (KAT)**. Our framework transforms:

- visual observations into a *sequence of 3D visual keypoints*, which are then tokenised as strings of characters: we define those as **keypoint tokens** (Fig. 7.3).
- end-effector trajectories of $SE(3)$ poses, into a *sequence of 3D points* fixed in the end-effector frame, which are also tokenised into a sequence of characters: we refer to those as **action tokens** (Fig. 7.4).

Large Language Models process text as sequences of tokens from a fixed vocabulary. To input numerical keypoint and action data, we must convert floating-point coordinates to text strings. Our tokenisation procedure is as follows:

Step 1: Coordinate quantisation. We first convert floating-point coordinates (in meters) to integers (in millimeters) by multiplying by 1000 and rounding: `x_int = round(x_float * 1000) # e.g., 0.342 meters → 342 mm`

Step 2: String formatting. Each coordinate is converted to a string representation with consistent formatting: `x_str = str(x_int) # e.g., 342 → "342"`

Step 3: Sequence construction. Keypoints and action tokens are concatenated with delimiters to form the final prompt. For a keypoint $k = (x, y, z)$:

`keypoint_str = f"(x_str,y_str,z_str)" # e.g., "(342,156,89)"`

We will now describe the processes of obtaining the keypoint and action tokens in more detail.

Keypoint Tokens

The observations received by the robot are in the form of top-down RGBD images o_i of dimension $(512, 512, 4)$, taken from a calibrated RGBD camera at the beginning of each episode. To accelerate the learning process, we aim to extract lower-dimensional data from the sequence of pixels. In particular, we extract from each image o_i a sequence of *visual keypoints* $k_{1:K}$, as illustrated in Fig. 7.3. To do so, we leverage recent Vision Foundation Models: large vision networks pretrained on vast datasets of images. We use DINO Vision Transformers (DINO-ViTs) [30], a family of networks that has been shown to be able to extract information-rich descriptors from images, representing both the semantic and geometrical information present in the image. We follow the method proposed in [9]: we first compute the dense descriptor tensors E of shape $(H, W, 6528)$ (where $H, W < 512$ and depend on the patch size selected) from each image collected during the demonstrations in $\mathcal{D}$, by inputting each image into the DINO-ViT. We then extract, through a Best Buddies Nearest Neighbours method [9], a sequence of K descriptors, e.g. one-dimensional vectors of shape $(1, 1, 6528)$ corresponding to a particular patch

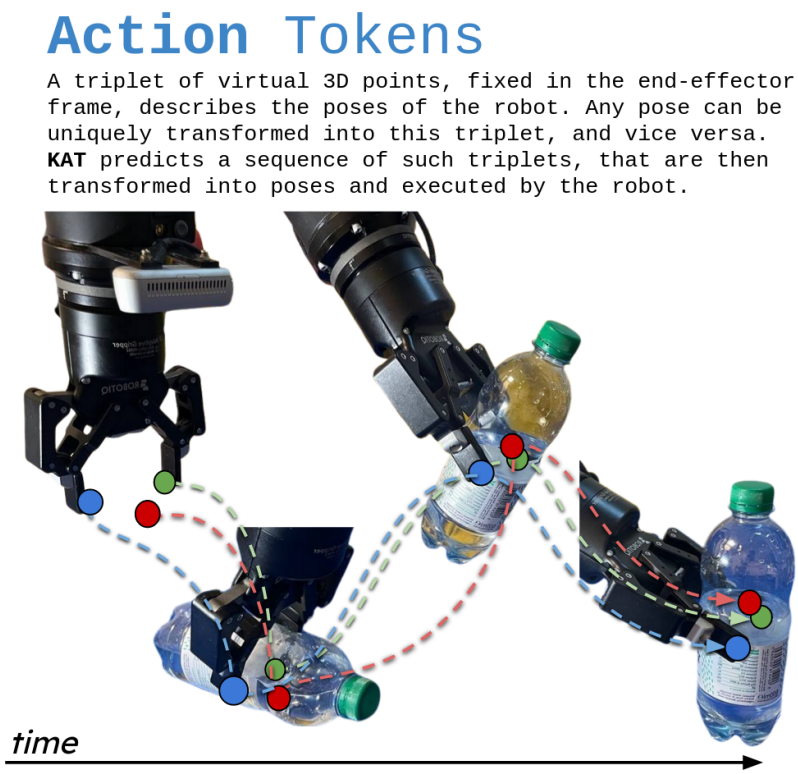


Figure 7.4: Illustration of the *action tokens*, used in this work to represent $SE(3)$ end-effector poses.

in one of the demonstration images among the $H \times W$ patches and corresponding descriptors. Intuitively, the extracted descriptors represent parts of the image that are common to all the collected observations, therefore encoding a particular semantic or geometrical structure (*e.g. the handle of a mug, the cap of a bottle*). This process is entirely autonomous. Once we have obtained these K descriptors, we can extract the visual keypoints from test time observations o_t. To do so, we extract the dense descriptor tensor E_t from o_t, and find the closest descriptor to each of the K stored descriptors via nearest neighbour search, obtaining K 2D spatial locations h_k, w_k. These represent 2D keypoints in the image. Through the calibrated RGBD camera, we can first obtain the depth component of the 2D pixels, making them 3D, and then transform them into world frame $\mathcal{W}$ coordinates from camera coordinates. The final output of this process is a *sequence of 3D visual keypoints in world coordinates*. By representing these numbers as characters we obtain the **keypoint tokens** k, which will then be used as the input sequence $\mathcal{O}$.

Action Tokens

Given input $\mathcal{O}$, the human experts provide a sequence of actions $\mathcal{A}$, that is a trajectory of end-effector poses, which demonstrates a desired behaviour. In our work, we represent end-

effector poses as a triplet of 3D virtual points expressed in meters $e_1 = [0.05, 0.0, 0.0], e_2 = [-0.05, 0.0, 0.0], e_3 = [0.0, 0.1, 0.0]$, represented collectively as $e = [e_1^T, e_2^T, e_3^T]$, defined in the end-effector frame $\mathcal{E}$. Each end-effector pose is then represented as the triplet of points in world frame $e^{\mathcal{W}} = \mathcal{T}_{\mathcal{E}}^{\mathcal{W}} e$, where $\mathcal{T}_{\mathcal{E}}^{\mathcal{W}}$ is obtained through the robot's forward kinematics. The mapping is unique and invertible: each pose can be uniquely represented as a triplet $e^{\mathcal{W}}$, and vice versa, from each triplet we can recover the corresponding pose.

Note that representing an $SE(3)$ pose as a triplet of 3D points is mathematically *redundant*. Why did we opt for this redundancy? The reason is that, in this way, each 3D point lies in the same Euclidean space of the 3D keypoints tokens, therefore assisting with pattern recognition. On the other hand, the use of rotational representations, like angle-axis, Euler angles or quaternions, while requiring fewer numbers, would have required to implicitly learn a mapping from a Euclidean space to a different manifold. In our Experiments section we will empirically prove these benefits.

In Fig. 7.4 we illustrate a trajectory of the end-effector placing a bottle upright, showing the triplet of points moving in space. Each triplet $e^{\mathcal{W}}$ is represented by 9 numbers, which we can represent as characters to obtain **action tokens** a. A trajectory $\mathcal{A}$ is therefore represented as a sequence of action tokens.

While the *action tokens* computed from the demonstrations always form a triangle with a predefined shape, these tokens are independently predicted at test time and so this geometrical constraint may not always hold. We therefore find the closest triangle via a Least Squared Error approach, and from that compute the end-effector pose.

To represent the gripper status, we add an additional number δ_g to the 9 representing the pose, indicating an open gripper if $\delta_g = 0$ and a closed gripper if $\delta_g = 1$. At test time, we round possibly noisy predictions as $\delta_g \geq 0.5 \rightarrow \delta_g = 1$, and $\delta_g < 0.5 \rightarrow \delta_g = 0$. This gripper representation is used in the same way in all the baselines we will later introduce.

This representation is similar to the one proposed in [113], studying efficient representations for grasping, that however operates in pixel-space: they in-paint a triplet of pixels on the visual observation representing where the end-effector would move. They then sample possible pixel-locations of these triplets, and use a learnt network to verify if that would lead to a successful grasp. Our representation is instead in 3D world-frame Euclidean coordinates, and we output a sequence of such poses to perform tasks well beyond grasping.

In-Context Imitation Learning via Pretrained Transformers

Transformers trained on textual datasets to perform language modelling, referred to as Large Language Models, receive a *prompt* p_i as an input, that is a sequence of tokens, and autoregressively compute a new sequence of tokens $t_i \leftarrow f(p_i)$ that maximise the likelihood $p(t_i|p_i)$. While originally trained to complete textual prompt, and therefore to produce coherent new text given a prompt emulating the natural language probability distribution appearing in the data, these Transformers were observed to gain the emergent capability to perform general pattern completion well beyond the domain of natural language [142]. In each training episode the robot collects a visual observation and a trajectory of actions recorded by the expert demonstrator: the observation is transformed into a *sequence of keypoints tokens* and the actions are transformed into our chosen triplet representation as a *sequence of action tokens*. At test time, the robot collects an observation that is also transformed into a new *sequence of tokens*. When receiving these sequences of keypoint and action tokens as a prompt p_i, the Transformer therefore will be conditioned to output, given the novel sequence of keypoint tokens, a new sequence of action tokens that mimics the behaviour demonstrated by the expert in $\mathcal{D}$. The action tokens, i.e. the triplets e^W, will then be transformed into $SE(3)$ poses of the end-effector and executed sequentially.

As such, in this work we tackle the sequence-to-sequence few-shot imitation learning problem by first transforming all sequences in $\mathcal{D}$ into keypoint action tokens $[k_d, a_d]_{d=1}^D$, appending each one to the list of tokens p_t given as input to the Transformer. We also represent the new observation o_t for the task at hand as a sequence of keypoint tokens k_t, and append those to p_t as well. The robot will then execute the sequence of predicted action tokens $a_t \leftarrow f(p_t)$ with $p_t = [[k_d, a_d]_{d=1}^D, k_t]$, transformed into a trajectory of $SE(3)$ poses.

The advantages of using a large pretrained Transformer as an *imitation learning machine* are twofold:

- **Data efficiency**: the vast pretraining allows the Transformer to **more efficiently extract patterns from sequences** than training policy networks on that data from scratch.

- **Time efficiency**: *in-context learning* only requires a single forward pass through the network and no weight updates. This results in the ability to **deploy the robot immediately after receiving the demonstrations**, in contrast to the need to finetune networks on the new data.



Figure 7.5: The tasks we evaluated our method and the baselines on.

7.4 Experiments

Our work proposes the use of text-pretrained Transformers as few-shot imitation learning machines with our Keypoint Action Tokens framework. The core investigation performed in our experiments is therefore aimed at answering two main questions: **1)** How does **KAT** compares against current state-of-the-art imitation learning frameworks? **2)** What is the optimal design of keypoint and action tokens in order to maximise performance, and why?

Additionally, we also provide a series of ablation studies and analyses designed to shed light on more properties of our method, like robustness to distractors.

Experimental Setup

We run our experiments using a Sawyer robot with 7 degrees of freedom, interacting with objects on a table in front of it. The end-effector is a Robotiq 2F-85, on which we mount an Intel Realsense D435 RGBD camera. The robot is controlled via position control. In every interaction episode, the robot records one visual observation at the beginning of the task, then computes a trajectory of actions, and finally executes those actions. The end-effector is moved 90cm above the table at the beginning of each episode to record an image observation.

During demo collection, the robot first captures an observation, and then the human expert manoeuvres the end-effector through kinesthetic teaching, while its $SE(3)$ poses are recorded. Each demo therefore collects an RGBD observation and a sequence of end-effector poses. When collecting demos, the positions of the objects for each task are randomised from within the starting state distribution of the task.

At test time, the objects are positioned on the table in novel configurations, designed to test both *interpolation* and *extrapolation* abilities with respect to object poses during training. The robot captures an observation as above, and then computes a series of end-effector poses.

We use **GPT-4 Turbo** [61] as the Large Language Model, but later compare its performance to other models.

Method/Task	T	Wipe	Sweep	Espr.	Scoop	Pick & Pour	Hang	Bottle	P. & Place	Avg.
KAT (Ours)	**0.8**	**0.8**	0.7	**0.4**	**0.6**	**0.7**	**0.7**	**0.6**	**0.8**	**0.68 [0.58, 0.77]**
Diff. Policies	0.0	0.2	0.1	0.0	0.0	0.0	0.2	0.1	0.3	0.1 [0.05, 0.18]
KeyAct-DP	0.4	0.7	**0.8**	**0.4**	0.4	**0.7**	**0.7**	0.4	**0.8**	0.59 [0.49, 0.69]
KeyAct-MLP	0.3	0.4	0.7	0.3	0.4	0.6	**0.7**	0.4	0.7	0.5 [0.40, 0.60]

Table 7.1: Success rates with 10 demonstrations for each task, for KAT and the baselines (best in bold). We also provide 95% confidence Wilson scores on the aggregated average success rates.

Tasks

We thoroughly evaluate our method by choosing a family of everyday manipulation tasks and measuring its ability to efficiently replicate the expert behaviour, generalising after just a few demonstrations. The criteria we adopted to choose the tasks are: **1)** to replicate tasks that appeared in the recent literature, **2)** to measure **KAT**'s ability to tackle a series of challenges that appear in everyday tasks. In particular, we chose tasks that collectively measure **generalisation to novel shapes, multi-modality, precision, dexterity, multi-stage execution, and 6D trajectories**. The tasks are the following:

- **Align T**: Inspired by [35], the robot needs to move a T-shaped object to align it to the axes of the table. The task requires *non-prehensile* abilities and the ability to model *multi-modal* distributions: demos are provided following different strategies (pushing the vertical or the horizontal part to align the T), and the robot needs to commit to a single one at test time.

- **Wiping a Plate**: The robot needs to follow the edge of a plate with a sponge in a circular motion. This task requires *multi-modal* reasoning: half the demos are given in clockwise direction, half are given counter-clockwise. The robot needs to commit to one at test time. We test for both interpolation and extrapolation abilities by also using unseen plates at test time of unseen dimensions.

- **Sweep**: The robot needs to sweep an object into a dustpan. The relative positions of dustpan and object are randomised, therefore the robot needs to compute an effective trajectory to bring the object into the dustpan. We test for generalisation by also using a novel dustpan and objects to sweep at test time.

- **Espresso**: The robot needs to insert an espresso capsule into a toy espresso machine, and then close its lid. This task requires noteworthy *precision*, and is composed of *multiple stages*.

- **Scooping**: The robot needs to scoop some chocolate powder from a cup with a spoon.

The task was proposed in [220] as a challenging, dexterous task. We test for generalisation by using also unseen cups at test time.

- **Pick and Pour**: The robot needs to pick up a French Press and pour coffee into a cup. The pouring part of the task was proposed in [220] as a *challenging, dexterous task*. By also picking up the French Press, the task requires *multiple stages*. We test for generalisation by also using unseen cups at test time.

- **Hang**: The robot needs to hang a clothes hanger onto a horizontal support. The task requires to reach a *precise position and height*, otherwise the hook would not hang on the support.

- **Put Bottle Upright**: The robot needs to pick up a horizontally placed bottle and place it upright on the table. This is also inspired from a task in [35], where a mug has to be flipped. The task requires a *non-trivial, 6D trajectory* to be completed, so that the bottle is stably placed. We test for generalisation by also using unseen bottles at test time.

- **Pick and Place**: The robot needs to pick up an apple and place it in a red bowl, with a blue bowl acting as a visual distractor. We test for generalisation at test time by both using an orange instead of an apple, and a purple bowl as a distractor instead of the blue bowl. This task requires the ability to *ignore distractors* and understand which objects are useful, and which must be ignored.

Baselines

As the goal of our work is to demonstrate the efficiency of Large Language Models (large text-pretrained Transformers) to effectively act as *imitation learning machines* by learning to emulate expert behaviour after a few demonstrations, we compared our approach with Diffusion Policies [35], a state-of-the-art general imitation learning algorithm from the recent literature, that was already demonstrated to surpass a series of recent techniques in terms of learning efficiency and generality.

We compare to two versions of this. First, the original image-based **Diffusion Policies** method proposed in [35]. Second, a modified version were we provide our proposed *keypoints* as input and *actions representation* as outputs (**KeyAct-DP**), therefore going from a $(512, 512, 4)$ input space to K 3D visual keypoints, and from the original position and orientation representation of [35] to a triplet of 3D points uniquely representing an end-effector pose. We use the original authors' code, and optimise hyperparameters to maximise performance on our tasks.

Furthermore, we compare against using a feed-forward network to learn to map the same keypoints into a sequence of action tokens, (**KeyAct-MLP**).

These baselines allow us not only to compare **KAT** to the state-of-the-art in the field, but also to measure the contribution coming from the *keypoints* and *actions* representation we propose.

Collecting Demos and Train-Test Distributions

When collecting demos, we randomise the pose of the objects on the table. In particular, we move the objects on a $50cm \times 30cm$ area, and randomise their orientation around the imaginary vertical axis in a 45 degrees range from a "standard orientation". To test for extrapolation at test time, we move objects in a larger area of $70cm \times 40cm$ and rotate them in a 60 degrees range.

For tasks that can be tested for objects generalisation (*Wiping a Plate, Sweep, Scooping, Pick and Pour, Bottle Upright, Pick and Place* we provide demonstrations using 2 different sets of objects, and then test on both these sets and a third unseen set.

When testing for robustness to distractors, we change our background, which is normally a black cloth, to expose the wooden table underneath, and position an additional set of 2-3 random objects of different classes then the ones involved in the task, randomly positioned as the other objects.

Prompting the Language Model

We mainly use GPT-4 Turbo for our experiments, except when explicitly comparing it to GPT-3 and GPT-2. For GPT-4 and GPT-3 we use the official OpenAI API. For GPT-2, we use the Transformers library by HuggingFace to compute its outputs locally.

While largely receiving tokenised numbers as input, the way GPT-4 is structured requires a "*system*" message describing what its task is. We therefore briefly describe that GPT-4 is tasked to solve a time series pattern recognition task, receiving examples of sequence-to-sequence data, and needing to find an output sequence given an input sequence that emulates the underlying patterns of the inputs. Notice we *do not mention robotics or tasks anywhere*: the model is only asked to act as a sequence-to-sequence general pattern recognition machine.

In particular, the instruction is

"You are a pattern generator machine. I will give you a series of patterns with INPUTS and OUTPUTS as examples. Then, you will receive a new INPUTS, and you have to generate

OUTPUTS following the pattern that appears in the data. The points are (x,y,z) coordinates. Only reply with the OUTPUTS numbers."

When comparing Language Models, we use the same prompt for each, composed of the above, plus the tokenised demonstrations as sequences of keypoints and action tokens, and finally the test time sequence of new keypoint tokens.

Extracting descriptors from DINO, CLIP and R3M

Here we describe in more detail how we extract the dense descriptor maps, given an input observation, to then extract keypoints. For DINO, we use *DINO-ViTs8*, with a stride of 4. We extract the *key* descriptors from the 9th layer. For CLIP, we use the CLIP RN50 version, and extract the output of the fifth-to-last layer of the ResNet50 backbone, a 3 dimensional tensor. Likewise for R3M, we extract the output of the fifth-to-last layer of the ResNet50 backbone, a 3 dimensional tensor. We explored each of these hyperparameters to maximise the performance of each of these methods.

Results on Few-Shot Imitation Learning

Can a text-pretrained Transformer/Large Language Model be used to perform in-context imitation learning of expert behaviours? This is the core question we explored in this work. To answer this, we evaluated **KAT** and the baselines on the 9 tasks reported above. For each task, we provide **10 demonstrations and execute 10 test time episodes by randomising the position of the involved objects**. 5 episodes are tested with the same object(s) seen during training, while the remaining 5 are tested with *unseen objects* where possible (as listed above, the only exceptions being *Align T, Espresso* and *Hang*). We extract $K = 10$ *keypoint tokens* from the observation recorded at the beginning of the episode, and record the demonstrations actions at 4 Hz, therefore storing 4 poses per second. In later section, we will show the effect of these hyperparameters on the overall performance.

We show success rates in Table 7.1. These results provide a series of interesting insights. First, the analysis of success rates and Wilson intervals show that **KAT performs at the level of the state-of-the art in imitation learning, with better performance on certain tasks**. However, KAT does not need any training after receiving the demos, unlike the baselines that require $\sim$10 minutes of training per task. Second, our novel representation of observations and actions as *3D points*, dramatically improves the performance of both Diffusion Policies over the original image-based Diffusion Policy method, and also make a simple MLP baseline perform

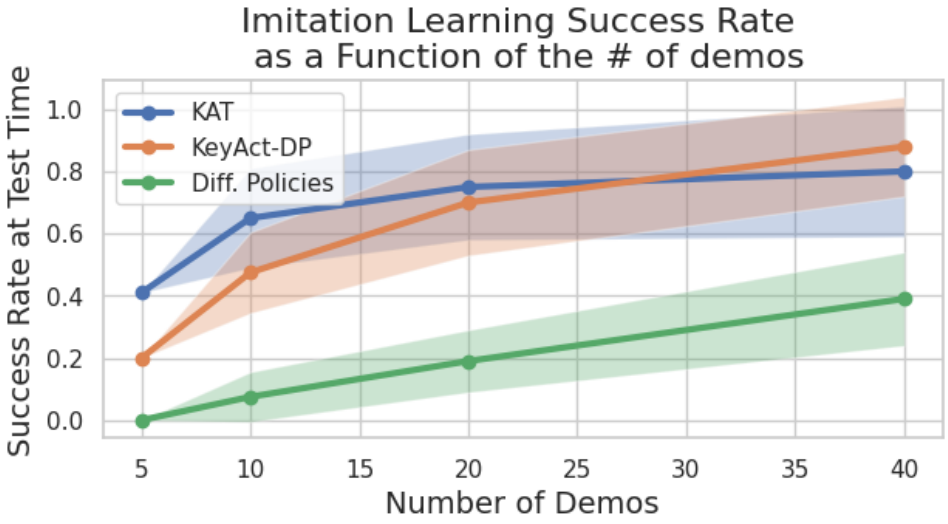


Figure 7.6: Success rate of each method as a function of the number of demos. While KAT outperforms the baselines in the few-shot regime ($\leq$ 20 demos), in-context learning struggles to improve as the number of demos increase even more. Plot shows mean and standard deviation across tasks.

better than end-to-end Diffusion Policies. Qualitatively, we also observed that Diffusion Policies struggled more to extrapolate to regions of the environment or object not covered during the training process.

How does the performance change as the number of demonstrations increases? In our previous experiments we provided 10 demonstrations for each task. To investigate the performance as a function of the number of demonstrations received, a key property in few-shot imitation learning, we chose a subset of the 9 tasks (*Align T, Wipe Plate, Espresso, Bottle Upright*) and provided 5, 10, 20 and 40 demonstrations for each task. We then compared the performance of KAT and the baselines. The results are plotted in Fig. 7.6. Diffusion Policy improves with the number of demonstrations, but at a very low rate and performance is poor even with 20 demonstrations, as the input space is substantially high-dimensional. Both KAT and KeyAct-DP achieve very good performance, with KAT's performance evolving more quickly than KeyAct-DP. However, when receiving 40 demonstrations, we can see that KeyAct-DP surpasses KAT on certain tasks. This suggests an important insight: pretrained Transformers can excel at in-context learning of sequence-to-sequence patterns few-shot, making them particularly useful for low-data regimes, but currently do not scale well as further in-context data is provided. We hypothesise that with this amount of data, in-context learning stops working optimally, and becomes a bottleneck. Instead, we believe that, given this amount of data, it may be more appropriate for Transformers to be finetuned to further improve performance, albeit the process would be more laborious that in-context learning. Nevertheless, finetuning

	Clean	Distractors	Diff. BG	Both
KAT (Ours)	0.65	0.6	0.62	0.58

Table 7.2: Success rates for our method showing the effect of distractors and different background (BG).

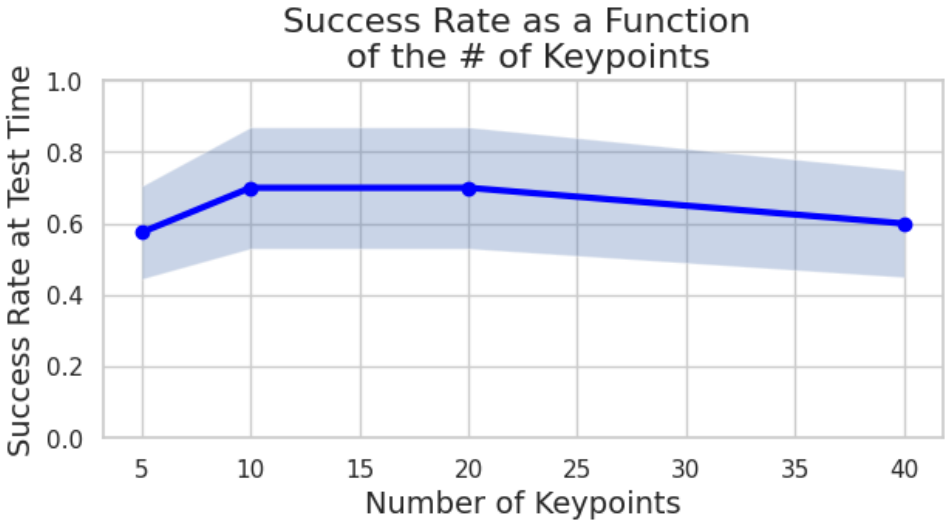


Figure 7.7: Success rate of our method as a function of the number of keypoints extracted. Plot shows mean and standard deviation across tasks.

LLMs on Keypoint Action Tokens data is an interesting direction for future work.

Can our method handle visual distractors? To study robustness to distractors, we evaluate KAT on the *Align T, Wipe Plate, Espresso, Bottle Upright* tasks, providing 10 demonstrations per task, but at test time we add distractor objects, change the background, or both. For each of these scenarios, we run 10 test time episodes, reporting the results in Table 7.2. These results showcase the robustness of KAT to visual distractors. This ability is obtained as a combination of two factors: from a vision perspective, the descriptors extracted from the observations via DINO-ViTs are robust to perturbations and have high dissimilarity across different objects. From a sequence-to-sequence pattern learning perspective, the pretrained Transformer is able to infer the *keypoints tokens* most relevant to the task by comparing the inputs-outputs relations in the demonstrations, therefore ignoring possible *keypoint tokens* that are less robust to unrelated visual changes.

Vision: Investigations on Keypoint Tokens

What is the optimal number of keypoints to extract? In this work we proposed a pipeline, based on the DINO family of Vision Foundation Models, to transform a visual observation o_i into a *sequence of 3D keypoints* $k_{1:K}$. The number of extracted keypoints K is an influential hyperparameters: a small K reduces input dimensionality, but generates a strong

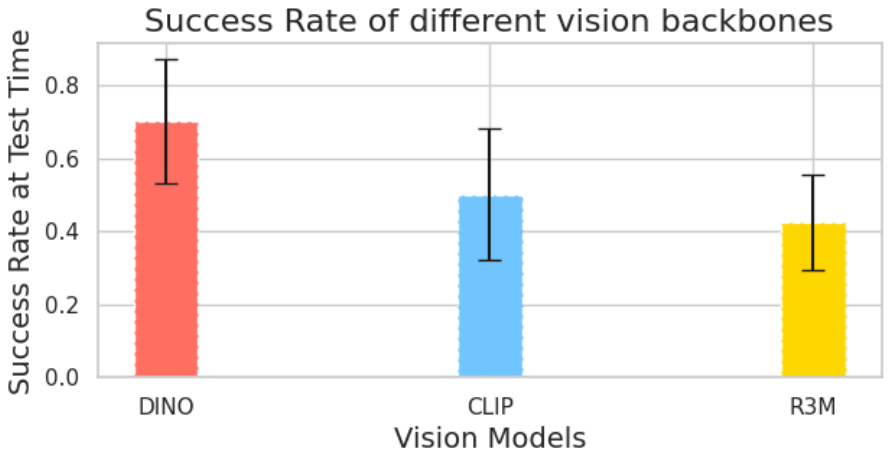


Figure 7.8: Success rate of our method when changing the vision backbone model from which we extract *keypoint tokens*. Plot shows mean and standard deviation across tasks.

information bottleneck and may fail to capture more nuanced geometric and semantic information from visual observations. On the other hand, a large K allows us to capture many visual details, but it is then more susceptible to noise and visual distractors. We measured the performance of KAT on a subset of the 9 tasks (*Align T, Wipe Plate, Espresso, Pick and Place*) as a function of K, providing 10 demonstrations per task. We repeat the experimental procedure described in Sec. 7.4 4 times, and extract $K \in \{5, 10, 20, 40\}$ keypoints from the observations, varying it at each experiment. We measure the performance on 10 test time episodes, and report mean and standard deviation in the plot of Fig. 7.8. The results show that the optimal K lies between 10 and 20, with minimal variations in that range.

What is the best vision model to extract descriptors from? To motivate our choice of using DINO-ViTs as vision backbones, we compared them to two other popular models in the robotics and computer vision literature: **CLIP** [162] and **R3M** [147]. We apply the same algorithm described in Fig. 7.3, with the difference that the dense descriptor tensors are extracted from CLIP or R3M. These experiments allowed us to test the robustness and representation abilities of these hidden representations. We run the same experimental setup described above in the previous subsection, setting $K = 10$ and training and testing on *Align T, Wipe Plate, Espresso, Pick and Place*. Our results demonstrate that DINO is the optimal choice, coherently with the results from the recent literature [9]. While R3M is pretrained on data more related to robotics (first person videos of humans performing manipulation tasks [59]), interestingly it does not perform better than DINO, pretrained on a large task-agnostic dataset of web images [30]. This apparently counter-intuitive result was also reported in [43].

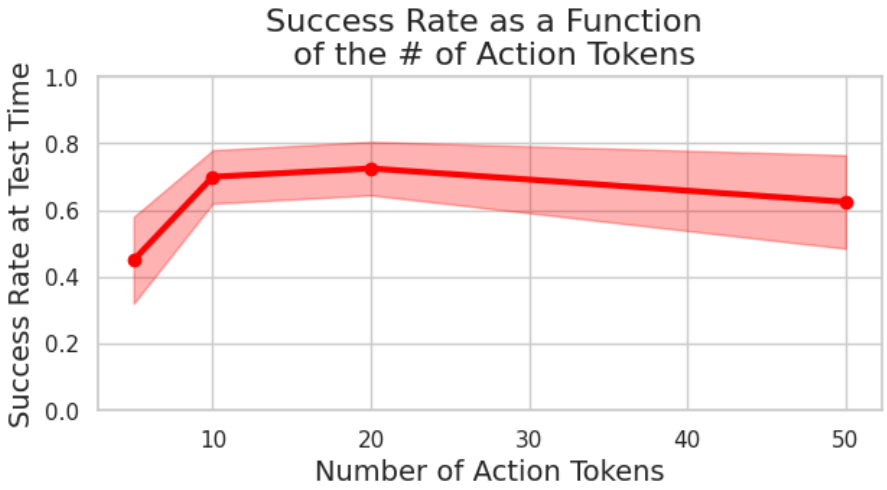


Figure 7.9: Success rate of our method as a function of the number of action tokens extracted from each demonstration trajectory, that strongly correlates with the number of action tokens generated at test time. Plot shows mean and standard deviation across tasks.

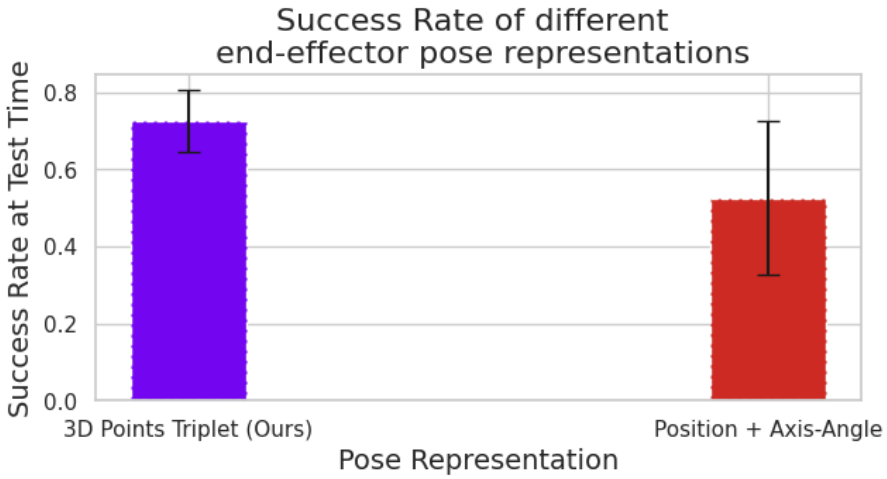


Figure 7.10: Success rate of our method with different representations of the end-effector poses. Plot shows mean and standard deviation across tasks.

Action: Investigations on Action Tokens

What is the optimal number of actions to record and predict? In our work, actions are recorded as end-effector poses during demonstrations, then tokenised into *action tokens*, each being a triplet of 3D positions in the world frame. Vice versa, at test time the Transformer predicts a series of *action tokens* that are then transformed into end-effector poses and executed. An LLM Transformer is able to autonomously decide to stop the autoregressive generation of tokens, by generating an internal STOP token. The number of action tokens it predicts is often very close to the number of action tokens each demonstration was composed of.

Therefore, we investigate what is the optimal number of action tokens N_a to provide for each demonstration, to guide the test time generation. A small N_a reduces the length of

the sequence and makes autoregressive generation easier and less prone to errors: however, it can also excessively subsample the trajectory of the end-effector, losing subtle but important movements. On the other hand, while a large N_a allows to precisely capture movements, it also makes both the input sequences excessively long and difficult to process, and also the generation phase prone to errors, as common with sequence-to-sequence networks tasked to predict very long outputs.

We replicate the experimental setup of Sec. 7.4. We record 10 demos on the *Align T, Wipe Plate, Sweep, Bottle Upright* tasks at a very high frequency, and then subsample those to change the number of action tokens N_a each demonstration is decomposed into, with $N_a \in 5, 10, 20, 50$. We then test the performance of KAT by providing to the Transformer these different quantities of tokenised actions extracted from the demonstrations, running 10 test time episodes per task.

Our results, in Fig. 7.9, show that the optimal number of tokens lies around 20: however, the drop at 50 is not large, suggesting the Transformer is still able to process this sequence length.

What is the optimal representation of poses to use as action tokens? In Sec. 7.3 we introduced the way we tokenise end-effector poses, transforming them through an invertible mapping into a triplet of 3D points $e^{\mathcal{W}}$. Is this the most effective representation to represent, process and generate sequences of actions? To motivate our design choice, we compared our tokenisation pipeline to the representation of poses as 3D position and axis-angle representation for orientation. This representation is therefore composed of 6 numbers instead of 9, but the orientation part lies in a different, non-Euclidean manifold. As described in Sec. 7.3, all representations share an additional number δ_g representing the gripper open/closed state.

We replicate the above experimental setup, but compare the effect of representing and generating actions as action tokens $e^{\mathcal{W}}$ or as position and axis-angle representation. Results in Figure 7.10 demonstrate that our choice outperforms the more classical position/axis-angle method of representing $SE(3)$ poses. As discussed in Sec. 7.3, we hypothesise that representing poses as a set of Euclidean entities in the same frame as the keypoint tokens facilitates generalisation for learnt imitation learning policies, that instead struggle more to generalise and interpolate/extrapolate correct axis-angle representations beyond the instances of the tasks observed during the demonstrations.

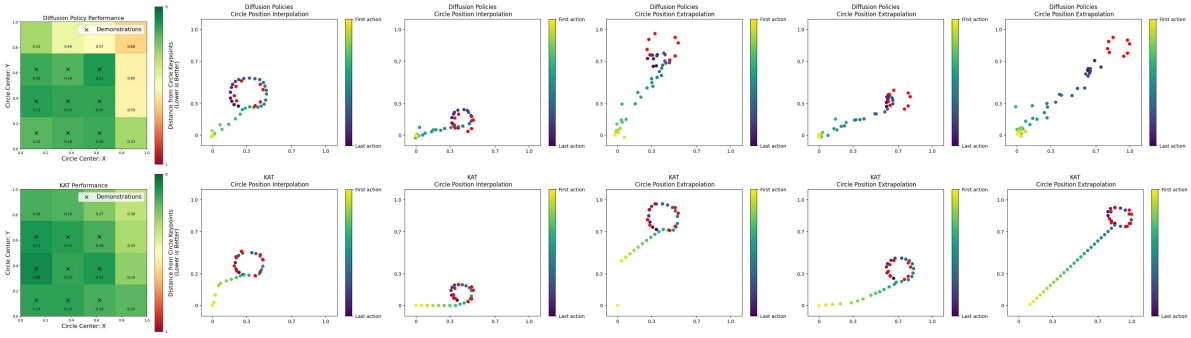


Figure 7.11: Detailed comparison of KAT and Diffusion Policies interpolation and extrapolation abilities on a 2D *draw circle* task. On the left, we show where the 9 demos were given, and binned performance on 100 test cases. On the right, we show a few example test cases, demonstrating also visually the difference in the trajectories generated by KAT and Diffusion Policies. In red we show the keypoints extracted from the randomly sampled circle. The trajectories are colored from yellow to green.

A Quantitative Analysis of Interpolation and Extrapolation Abilities between KAT and Diffusion Policies

In this section we study in more depth the abilities of KAT and Diffusion Policies to interpolate and extrapolate by conducted a more controlled study. To do so, we implement a new simulated 2D task. In this task, a circle is randomly sampled at the beginning of the episode, by randomly sampling its centre $c = (x_c, y_c)$ in the range $x_c \in [0.0, 0.66], y_c \in [0.0, 0.66]$, and its radius r_c in the range $r_c \in [0.04, 0.15]$. We then compute a list of keypoints from its circumference at pre-defined angles, $k_c = [(x_0, y_0), \ldots, (x_C, y_C)]$. These keypoints are the inputs received by the policy in this task. The outputs are a trajectory of 2D points $p_i = (x_{p,i}, y_{p,i})$. We provide 9 demonstrations where we manually draw a list of these points $[p_0, p_1, \ldots, p_D]$ starting in the bottom left corner and drawing a circle following the keypoints. We then train KAT and a Diffusion Policy to emulate this behaviour: when receiving a new set of keypoints, their goal is to generate a trajectory of points that draw that circle. A few visual examples are provided in Figure 7.11.

We use this simulated environment as it allows us to numerically evaluate the interpolation and extrapolation ability of the two methods. In particular, we sample 100 random circle and predict a trajectory using both methods, comparing how close they are to the desired circle to draw. The circles can also have their centre outside of the training distribution, in the range $x_c \in (0.66, 1], y_c \in (0.66, 1]$. We measure the error of a generated trajectory as the sum of distances between each circle keypoint k_c and the closest point from the trajectory, therefore

Method/Type of Error (low is better)	Interpol.	Extrapol.	Tot.
Diffusion Policy	0.16	0.51	0.314
KAT	**0.137**	**0.27**	**0.192**

Table 7.3: Numerical error on *draw circle* task of Diffusion Policy and KAT, highlighting interpolation, extrapolation and general performance. Lower values are better.

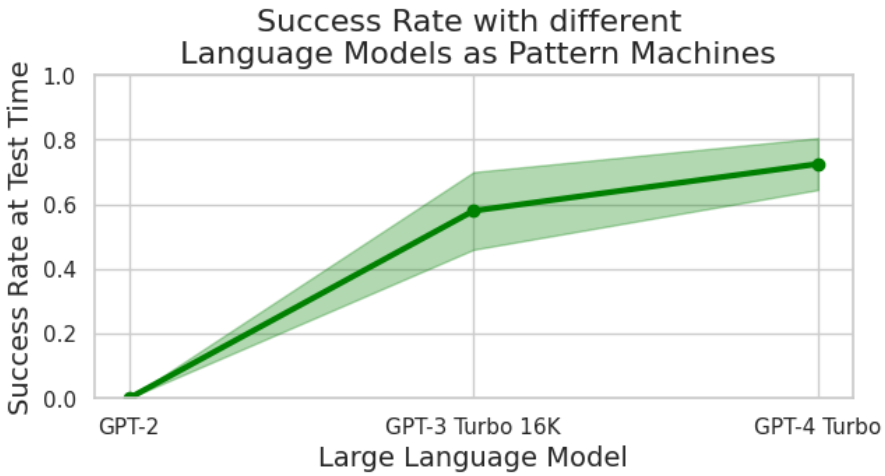


Figure 7.12: Success rate of KAT by varying the underlying LLM acting as the imitation learning machine. Plot shows mean and standard deviation across tasks.

a lower value means a trajectory that more closely follows the circle. In Figure 7.11 (Left) we group the performance of both methods into different areas of the input space: there is a clear drop in performance for Diffusion Policies when dealing with out of distribution inputs, as it can be seen by the score lowering on the top and on the right. Examples on the right side of the figure also visually show how the trajectories generated by Diffusion Policies tend to deteriorate the more the circle is distant from the demonstrations distribution. On the other hand, importantly, KAT's performance remains strong in all the input space, and the examples show how the trajectories are smooth and precise also when the circles are far from the training distribution. We also group the errors from both methods' trajectories in Table 7.3, clearly showing KAT's strong extrapolation abilities.

This study numerically proves how the use of Large Language Models as imitation learning machines allows us to leverage their unprecedented ability to extract patterns from few examples.

Better Imitation Learning Machines by Scaling Language Models

Our work is inspired and designed around the finding that large Transformers, pretrained on sequence-to-sequence tasks, have the emergent capability to infer in few-shot, patterns belong-

ing to different data modalities. This effect has been observed especially in Large Language Models. Scaling laws and empirical results have been obtained [93, 29] that allow the predict the performance of these models as a function of the size of both their parameters and training dataset.

As a result, the latest LLMs, like GPT-4 Turbo, can effectively act as few-shot imitation learning machines for robotics, reaching or surpassing the performance of state-of-the-art imitation techniques despite not needing any robotics data. As the field of LLMs has seen exponential growth in recent years [29, 58, 61, 57], we hypothesise that future LLMs will be even better, considering also recent breakthroughs in scaling reasoning models efficiently [44], leading to even more efficient imitation learning machines. And this is without the need for any particular innovation in robotics itself, be it in terms of data or algorithms. To concretely support this hypothesis, we now evaluate and visualise the performance of different generations of LLMs.

We test KAT on the *Align T, Wipe Plate, Sweep, Bottle Upright* tasks, selecting $K = 10$ and $N_a = 20$, but change the model of the LLM computing the output sequence given the demonstration sequences and the input sequence. We test three generations of **GPT** models: **GPT-2 [161], GPT-3 Turbo 16k [29], GPT-4 Turbo [61]**. Results in Fig. 7.12 demonstrate that the evolution of these text-pretrained Transformers, albeit independent in any way from robotics, leads to an improvement in performance, as more recent models are more efficient at few-shot sequence-to-sequence pattern recognition and generation. This suggests that future models will be even more efficient, resulting in more effective *general imitation learning machines*. This is a remarkable phenomenon, considering that it does not require gathering more robotics-tailored data, nor does it require actively researching fundamentally novel imitation learning algorithms.

A noteworthy aspect is the ability to effectively process longer sequences as inputs: GPT-2 was limited at 1024 word tokens, while later models saw a 10x-30x increase in a few years. Scaling the context input size of text-pretrained Transformers can help make in-context learning of trajectories more effective, tackling the bottleneck we observed in Fig. 7.6.

7.5 Limitations

While achieving superior results in few-shot imitation learning with $\leq 10 - 20$ demos, KAT does not currently scale as well as Diffusion Policies. We hypothesise that this is due to KAT relying entirely on in-context imitation learning, which can start acting suboptimally when

the input sequences (the tokenised inputs and outputs of the demos, and the tokenised new observation) become excessively long. We believe that once > 50 demos are available, it may be beneficial to finetune the pretrained Transformers instead of relying entirely on feed-forward in-context learning. To tokenise the visual observations, we rely on extracting a predetermined amount of keypoints K. While we showed that this approach is effective, and even significantly improves the performance of end-to-end vision based Diffusion Policy, the method cannot adapt the number of extracted keypoints to different situations, where e.g. more objects are present. Future work into adaptable and dynamic keypoint extraction may improve the performance of keypoint-based methods, like KAT.

Given a prompt of length $\mathcal{L}$, the computational complexity of generating output tokens, based on the classic Transformer architecture [193], is $\mathcal{O}(\mathcal{L}^2)$. Therefore, current Transformers will scale poorly to larger datasets of demonstrations in scenarios where online control is fundamental. However, there are several research avenues currently researching an efficient alternative to classic Transformers [36] or proposing alternatives which are equally effective but linear in complexity [81].

7.6 Discussion

We introduced **Keypoint Action Tokens**, a framework that enables in-context imitation learning of human demonstrations, by repurposing large Transformers pretrained on text as general sequence-to-sequence learners. By tokenising both visual inputs and action outputs into a format suitable for text-based Transformers, we demonstrate that we can achieve state-of-the-art results in few-shot imitation learning on a series of challenging everyday tasks. We also analysed what design decisions result in the optimal performance of our method. Our work demonstrates that the progressive evolution of large Transformers pretrained on language, where data is abundant, can unexpectedly benefit the field of robot learning, where data is scarce.

The ability to perform in-context learning unlocks versatile ways of conditioning robots on new behaviours to perform, without the need to rely on pretraining a large behaviour model, that will then be mostly fixed in its set of weights. Inspired by these capabilities, in the next chapter we demonstrate we can condition robot behaviour on an ever-growing dataset of human videos, collected by users in their natural environment. This, as we will see, unlocks both a very efficient and intuitive way of training robots, and continual learning capabilities that go beyond fixed-dataset paradigms.

Chapter 8

R+X: Retrieval and Execution from Everyday Human Videos

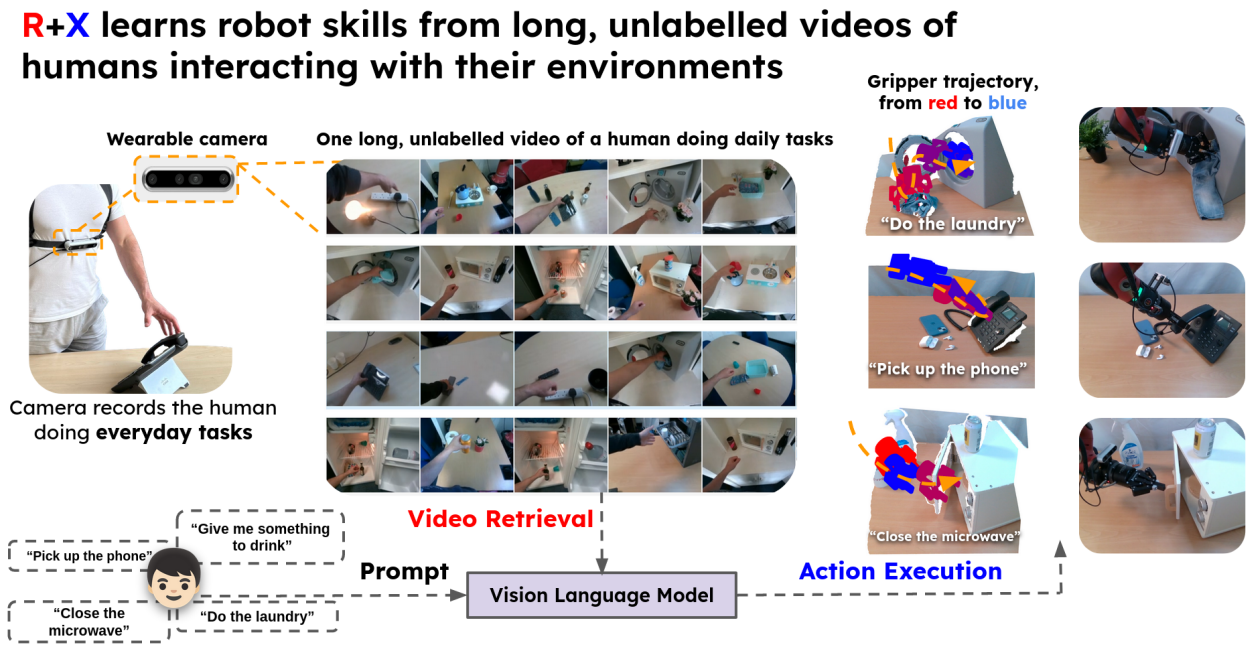


Figure 8.1: Given a language prompt, R+X first retrieves short relevant video clips extracted from a long unlabelled video of a human performing everyday tasks, recorded with a wearable camera. With the use of the retrieved video clips and a VLM, R+X performs in-context imitation learning allowing it to immediately generate and execute the desired behaviour on the robot. Videos are available on our website https://www.robot-learning.uk/r-plus-x.

8.1 Introduction

Robot learning of diverse, everyday tasks in natural environments, is a significant challenge. Imitation learning is a promising solution which has been widely adopted in recent years [39, 109], but it remains difficult to scale up due to the cost of hardware and human demonstration

time. If robots could instead learn from data which does not require any robot hardware or dedicated human time, the difficulty of scaling up data collection would be lowered substantially.

In this work, we study the problem of learning robot skills from long, unlabelled, first-person videos of humans performing everyday tasks in everyday environments. *"Long, unlabelled videos"* means that a human simply goes by their everyday life and passively records videos of themselves without the need to specify which behaviour is being performed. *"Everyday environments"* means that videos contain diversity in tasks, scenarios, objects, distractors, illumination, and camera viewpoints and motion. It is likely that such videos will become abundant through the adoption of wearable devices such as AR headsets and glasses [11, 141, 166, 136], and thus offer a significant opportunity for future scalability if robots could learn from such videos.

Previous approaches to learning from videos of humans have often relied on a set of strong constraints and assumptions, such as human videos manually aligned with robot videos, human videos manually labelled with language descriptions or demonstrations on robot or MoCap hardware [106, 17, 109, 91, 16, 222, 204]. In this work, we remove all of these constraints and present a framework that requires only an unlabelled first-person video depicting tens of tasks.

Our framework, **R+X**, is a two-stage pipeline of **R**etrieval and E**X**ecution shown in Figure 8.1 that uses Foundation Models for both stages. Upon receiving a language command from a user, a Vision Language Model retrieves all clips where the human executes the specified task. By extracting the trajectories of the human's hand in each clip, we then employ a few-shot in-context imitation learning method to condition on these trajectories, which enables a robot to ingest, learn from, and replicate the retrieved behaviours to previously unseen settings and objects. The recent literature demonstrated that, by finetuning large Vision Language Models on robotics data, they can transfer their common knowledge and ability to reason and plan to robotics settings [56, 52]. This, however, requires very expensive finetuning of often intractably large models. With our proposed framework, we can equally leverage these abilities but now via retrieval and video understanding, thus without the need for any finetuning. Rather than requiring explicit policy training on the retrieved videos, skills can be learnt, and executed immediately following the language command. In particular, we demonstrate the benefits that this brings over spatial and language generalization, compared to large monolithic policy networks.

In summary, the two main properties of R+X are: **1)** it enables robots to execute everyday tasks from language commands, given long, unlabelled, first-person videos collected naturally by recording a human's daily activities, and **2)** it achieves this without the need for any training or finetuning of models, allowing it to learn and execute tasks immediately.

	Multi task no label/align videos	No robot data	Non-prehensile tasks	New obj gener.	Distractors (both train & test)	No MoCap hardware
Vid2Robot	✗	✗	✓	✓	✓	✓
WHIRL	✗	✗	✓	✗	✓	✓
DITTO	✗	✓	✗	✗	✗	✓
ScrewMimic	✗	✗	✗	✓	✓	✓
Orion	✗	✓	✗	✗	✗	✓
DexCap	✗	✓	✓	✓	✓	✗
R+X	✓	✓	✓	✓	✓	✓

Figure 8.2: The main assumptions and constraints of many recent learning from observation methods, Vid2Robot [106], WHIRL [17], DITTO [91], ScrewMimic [16], Orion [222], DexCap [204]

8.2 Related Work

Collecting Robotics Data. Scaling data collection has been proven to be a successful path towards increasingly more general machine learning models [29, 57, 143, 94]. To collect robotics data, the most common paradigm is to teleoperate robots, collecting datasets of paired observations and actions [39]. This, however, needs dedicated teleoperation hardware, and needs human operators to allocate their time to *actively teach a robot new tasks*. In our framework, a human user interacts with their environments as usual, completing the tasks they wish, while a robot *passively learns to emulate it*, resulting in a more scalable and time efficient paradigm.

Learning from Human Videos. Many recent works have proposed solutions to teach new skills to robots by observing a human executing such tasks. However, as we illustrate in Fig. 8.2, they often relied on a set of assumptions and constraints that limited their use "in-the-wild". Bahl et al. [17], Wang et al. [203] require a combination of human data and either robot exploration or teleoperation, therefore needing active assistance of the user in teaching the robot, and videos are recorded from a fixed, third-person camera. R+X relies entirely on the videos recorded by the user with a mobile camera in their natural environments. Heppert et al. [91], Zhu et al. [222], Bahety et al. [16] can learn robot skills entirely from human videos. Heppert et al. [91], Zhu et al. [222], however, focus on replicating the object trajectory from the demo, and cannot perform tasks that do not involve grasping and moving, such as pushing or pressing, that R+X can execute. Bahety et al. [16] can learn a larger repertoire of skills, but still relies on learning single tasks in isolation from a fixed camera, while R+X can replicate tasks from a given a language command after receiving a single, long, unlabelled video depicting

tens of tasks, without the need to specify which clip demonstrated which behaviour. Methods like Wang et al. [204] allow a user to naturally interact with their environment while collecting data that can teach a robot new skills. However, they require additional hardware to wear, like specialised MoCap gloves. R+X only needs a single RGB-D camera, and does not require any extrinsics calibration. This means that data could be recorded using a wearable camera, smart glasses, AR visors, and more. Recently [20, 215] propose methods to learn skills from videos, similarly to our method, but focusing on optical flow between frames. [20] however learns a visual-goal-conditioned policy, while we focus on language-conditioned skills extracted from a long, unlabelled video. Additionally, by focusing on optical flow of relevant objects, these methods might struggle on objects that barely move for some tasks, like pressing, as we show we can do with R+X by tracking the human hand directly. [18] learns, from in-the-wild human videos, where objects can generally be grasped and in what directions they can be moved, allowing a robot to passively learn to push, pick up, pull, etc. However, their method is more limited in the amount of skills it can learn, focusing on mostly linear movement of objects, while we show we can learn more articulated and dexterous tasks. [33, 180] show that human videos can be used to learn to guide robot exploration by extracting a reward function from videos. In our method, we show we can effectively learn skill without any need for robot teleoperation or autonomous exploration.

Language-Conditioned Policy Learning. Robots should learn a wide repertoire of skills, and be able to execute many different tasks while deployed. Language is considered a viable way to instruct robots and guide their task execution [39, 52, 56]. The common approach is to train a large, language and image conditioned policy that can, at test time, receive user commands and camera observations [39]. This, however, presents some criticalities: training such models can require enormous computational effort [56], and unlike other machine learning applications, robots should learn new skills over time. Therefore, re-training or finetuning such networks should be avoided. R+X therefore does not train or finetune any networks, but takes advantage of pre-trained models able to retrieve examples of tasks [57], and to learn to emulate and execute new behaviour directly at deployment via in-context learning [47].

Retrieval in Robotics. Other papers have proposed the use of retrieval in robotics. Zhu et al. [221] retrieves directly from a bank of *policies* trained on robotics data. Du et al. [53] retrieves from an unlabelled data of robotics experience, not human videos, and needs an additional example at test time to guide the retrieval phase, while we leverage a language command alone. DINOBot, described in Chapter 6, retrieves a goal observation to then perform visual servoing and replay a pre-recorded demo. R+X leverages large Vision Language Models

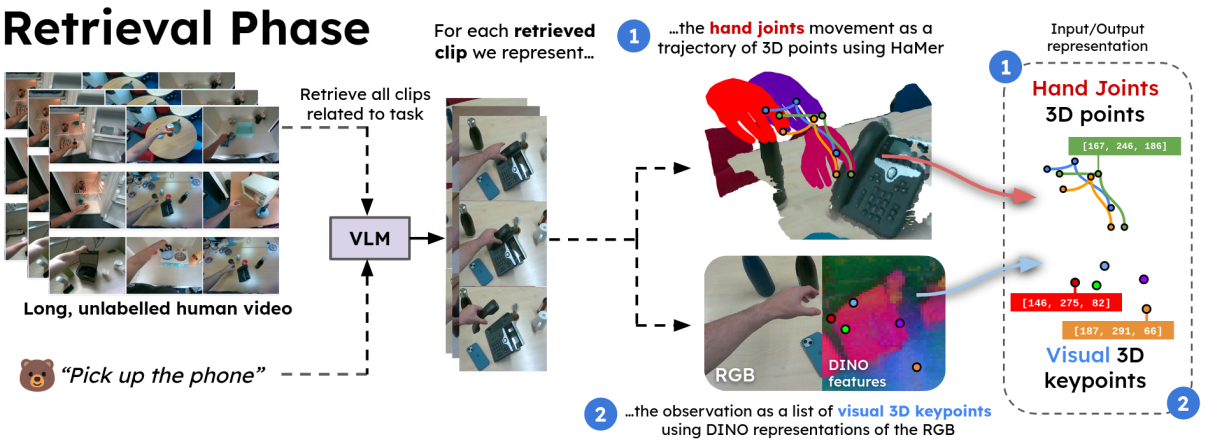


Figure 8.3: Upon receiving a language command, R+X retrieves all the relevant clips from the human video. Each retrieved clip is transformed from pixels to a sparse 3D points representations of the hand joints movement and salient parts of the visual observation.

to retrieve directly from long videos given a language query, without requiring pre-trained policies or new demonstrations as queries.

Retrieval Augmented Generation in Language Modelling: Large Language Models, through extensive pre-training and finetuning on web-scale datasets, are able to implicitly record countless facts and concepts in their weights [61, 151, 57]. However, it has been proved useful to add to their implicit knowledge the ability to explicitly access datasets of information, either offline or through web search [24]. This allows models to be more factually correct. In this work, we do not deal with factuality or information retrieval, but we take inspiration from these ideas to form a sort of "retrieval augmented execution", which is the main idea behind R+X.

8.3 Method

We now describe R+X, our proposed method to learn robot skills from long, unlabelled videos of humans interacting with their environments. We assume R+X has access to a long, unlabelled video of a user performing a multitude of tasks in many different locations. We call this long, first-person video of everyday tasks the "*human video*" $\mathcal{H}$ in the rest of the chapter. The goal of our method is to learn to emulate the behaviours recorded in the human video upon receiving a language command from the user.

Overall, from a high-level perspective, our pipeline takes three inputs: **1)** a single, long, unlabelled video of all the interactions recorded by the humans, $\mathcal{H}$ **2)** a task to execute in language form, $\mathcal{L}$ and **3)** the current observation of the robot as an RGB-D image, $\mathcal{O}_{live}$. It then outputs a trajectory of 6-DoF gripper poses, which are executed by the robot to tackle

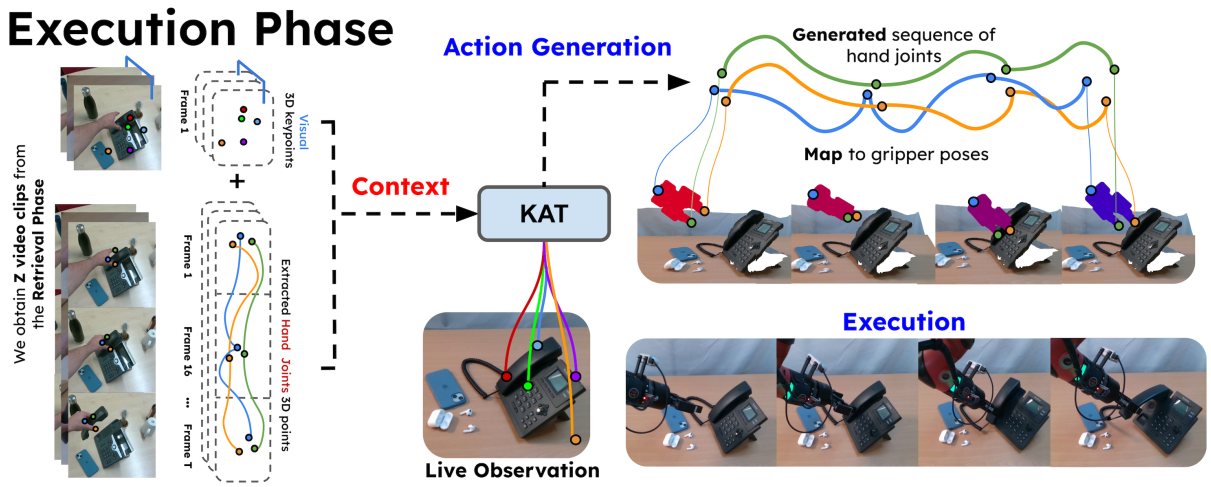


Figure 8.4: The visual 3D keypoints of the first frame of each of the Z videos obtained from the retrieval phase along with each extracted hand joint trajectory are used as context for KAT. To execute a skill, visual 3D keypoints are extracted from the live observation and used as input to KAT which generates a sequence of hand joints. By mapping this sequence to gripper poses the robot executes the desired task.

the task at hand.

There are however a set of non-trivial challenges: while receiving a language command to execute at deployment, the robot receives **no language information before deployment**: the recorded video contains no more information than the recorded visual frames. Additionally, the robot receives **no action information**: unlike the case of teleoperation, no joints or end-effector position/velocity/acceleration data are recorded, and the correct movements need to be inferred by the videos alone: the problem is additionally complex due to the cross-embodiment between human videos and final robot actions. Furthermore, as the user is interacting with their natural environment, the visual observations can be **filled with distractor objects** typical of household and offices, unrelated to the task the user is performing. To tackle these challenges, we leverage the abilities of Foundation Models.

Specifically, at the first phase of R+X, which we call the **retrieval phase**, we use a VLM to extract from the human video all the examples of the desired behaviour described in the prompt $\mathcal{L}$ as a list of Z shorter video clips $[\mathcal{V}_1, \mathcal{V}_2, \ldots, \mathcal{V}_Z]$. We map the Z videos into a lower dimensional 3D representation, extracting for each video: (1) a list of K *visual 3D keypoints* that describe the scene, $[k_1, k_2, \ldots, k_K]$ where $k = (x_k, y_k, z_k)$ (these are only extracted for the first frame of the video), and (2) the movement of the user's hand as a trajectory of length T of 21 hand joints that parametrise the MANO hand model [169], $\mathcal{J} = [j_1, j_2, \ldots, j_T]$, where $j = [[x_0, y_0, z_0], \ldots, [x_{21}, y_{21}, z_{21}]]$ (these are instead extracted for each of the T frames of the video). Finally, at the second stage of R+X, which we call the **execution phase**, to emulate the behaviours observed in the retrieved video clips, we condition a few-shot in-context imitation

learning model on this data that, given a live observation of the environment, it generates a trajectory of 3D hand joints to execute the desired behaviour described in the prompt. To map such joints to gripper poses, we designed a heuristic that we describe in later sections.

Retrieval: Extracting visual examples from a long, unlabelled video

The first main phase of R+X is **retrieval**, shown in Figure 8.3. Upon receiving a language command $\mathcal{L}$, and given the human video $\mathcal{H}$, the goal is to retrieve all video clips from the human video that depict the execution of the requested task. This is accomplished using a recent Vision Language Model (VLM), Gemini Pro 1.5 Flash [188]. Gemini, which we denote $\mathcal{G}$, is natively multi-modal and can take as input images, videos, and text, and outputs text. We prompt the model with the human video, and ask it to retrieve the starting and ending seconds at which the received task happens. The inputs of this phase is therefore a language command and the human video, and the output is a list of Z shorter video clips demonstrating the desired task, $\mathcal{G}(\mathcal{H}, \mathcal{L}) \rightarrow [\mathcal{V}_1, \ldots, \mathcal{V}_Z]$. Each clip comprises T RGB-D frames, where T can vary across different clips.

Preprocessing Videos into a Sparse 3D Representation

Given the Z extracted video clips, $[\mathcal{V}_1, \ldots, \mathcal{V}_Z]$, we apply a preprocessing step that converts each video clip from a list of RGB-D frames to a set of 3D points describing the visual scene and the trajectory of the human's hand, a representation that we will then feed to our few-shot in-context imitation learning model.

Visual 3D keypoints. To transform the complex RGB-D observations into a lower-dimensional, easier to interpret input, we harness the powerful vision representations generated by DINO [30], a recent Vision Foundation Model. As proposed in [9, 47], given the Z clips retrieved as described before, we find a set of K common 3D visual keypoints that capture interesting semantic or geometrical aspects of the scene. We extract these keypoints from the *first frame* of each of the Z videos only.

We first compute the DINO descriptors for the first

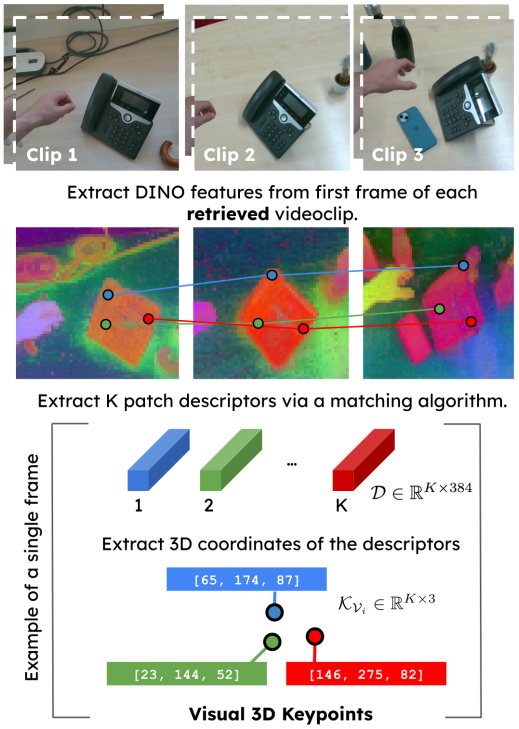


Figure 8.5: Visual keypoints are extracted autonomously from RGB via a matching algorithm [9].

frame of each of the Z videos, obtaining a list of Z different $N \times 384$ outputs, where N is the number of patches of each image [50, 61, 30]. Via a clustering and matching algorithm [9], we select from the $N \times 384$ descriptors of the first frame of the first video, $\mathcal{O}_{\mathcal{V}_1,1}$, a list of the K descriptors that are the most common in all the remaining $Z-1$ frames, as shown in Figure 8.5. We denote these descriptors as $\mathcal{D} \in \mathbb{R}^{K \times 384}$. Therefore, this way, we autonomously extract descriptors that focus on the object of interest, as its appearance is common among videos, while distractors and overall scene will vary [9, 47].

Finally, for each of the K descriptors in $\mathcal{D}$, we extract keypoints by finding the K nearest neighbours between the $N \times 384$ descriptors of each the Z frames, and compute their 2D coordinates. We then project these in 3D using each frame's depth image and known camera intrinsics. As such, given the retrieved clips, we obtain and store a list $\Lambda = [\mathcal{K}_{V_1}, \ldots, \mathcal{K}_{V_z}] \in \mathbb{R}^{Z \times K \times 3}$ of visual 3D keypoints, K for each clip, where each $\mathcal{K}_{V_i} = [k_1, ..., k_K]$ and $k_j = (x_j, y_j, z_j)$. For a more detailed description please refer to Amir et al. [9].

Hand Joint Actions. To extract human actions from each retrieved video clip we use the HaMeR model [156], a recent technique for 3D hand pose estimation from images based on DINO. In particular, using HaMeR we extract from each video frame $\mathcal{O}_{\mathcal{V}_z,t}$ (where $1 \leq t \leq T$) of each of the Z clips the 3D hand pose, represented as a set of 21 3D points, $j_{\mathcal{V}_z,t}$, describing the hand joints that parameterise the MANO hand model, as it is commonly done in the literature [169]. As the camera moves between frames, we design a stabilisation technique to compute transformations between camera poses that is robust to dynamic scenes. This enables us to express the extracted hand joints relative to a single reference frame, that of the first camera frame of each clip. For each video clip $\mathcal{V}_z$, this process results in a sequence of 3D hand joint actions $\mathcal{J}_{\mathcal{V}_z} = [j_{\mathcal{V}_z,1}, \ldots, j_{\mathcal{V}_z,T}]$, expressed relative to the first frame of each video $\mathcal{O}_{\mathcal{V}_z,1}$ where the visual 3D keypoints are also expressed in. As such, at the end of this process we are left with a list $\mathcal{M} = [\mathcal{J}_{\mathcal{V}_1}, ...\mathcal{J}_{\mathcal{V}_Z}]$ of Z hand joint action sequences.

In summary, from the Z retrieved video clips, we extracted the list of hand joints actions $\mathcal{M}$, along with the list of visual 3D keypoints Λ, that will be used as context for our in-context imitation learning model, as Z input-output pairs.

Execution: Few-Shot, In-Context Imitation from Video Examples

The second main phase of R+X is **execution**. Our framework is based on the use of a model capable of performing few-shot, in-context imitation learning, receiving a few examples of desired inputs and outputs pairs describing a desired behaviour, and able to replicate such behaviour immediately upon receiving a new input. We use **Keypoint Action Tokens (KAT)** to achieve

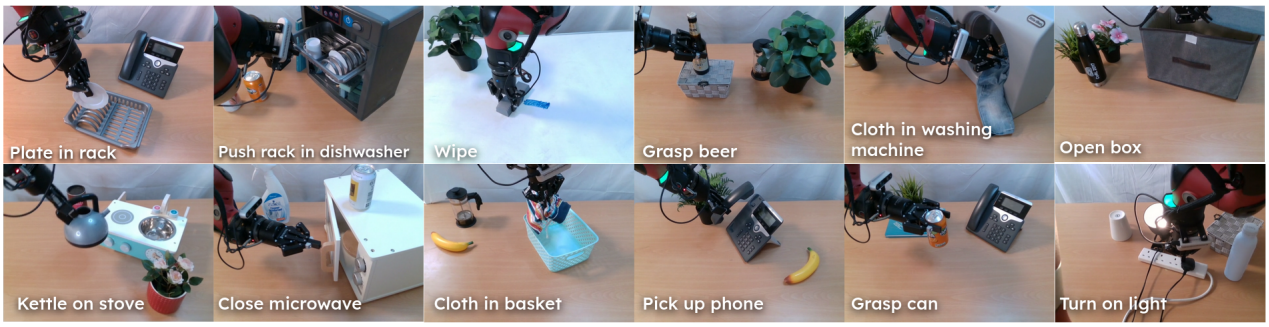


Figure 8.6: We test R+X on 12 everyday tasks, executed by a human user in different rooms and with different distractors.

this, a technique described in the previous chapter that takes 3D visual keypoints as input and outputs a trajectory of 3D points describing the gripper movement. Instead of explicitly training a model on robot data, KAT demonstrates that recent, off-the-shelf Large Language Models are able to extract such numerical patterns, and behave as few-shot, in-context imitation learning machines, without the need for any further finetuning.

Given the output of the retrieval phase $[\Lambda, \mathcal{M}]$ and a new, live RGB-D observation collected by the robot $\mathcal{O}_{live}$, we extract its visual 3D keypoints representation $\mathcal{K}_{live}$ by first extracting its $N \times 384$ DINO descriptors, and then finding the K nearest neighbours to each of the K descriptors in $\mathcal{D}$, that we obtained in the retrieval phase. This results into K 2D coordinates, that we project in 3D. We then input to KAT as context $[\Lambda, \mathcal{M}]$, as Z examples of the desired input-output mapping, and the new visual 3D keypoints, and generate a new trajectory of desired hand joint actions, $\text{KAT}([\Lambda, \mathcal{M}], \mathcal{K}_{live}) \rightarrow \mathcal{J}_{live}$ with $\mathcal{J}_{live} = [j_1, \ldots, j_T]$, as shown in Figure 8.4. We then map the predicted trajectories of hand joints actions into gripper poses, and execute them.

To summarize, given a language command $\mathcal{L}$ and a human video $\mathcal{H}$, R+X first **retrieves** Z videos depicting the described behaviour using a VLM. From the retrieved videos, a list of $[\Lambda, \mathcal{M}]$ visual 3D points and hand joint actions are extracted. Then, to execute the desired behaviour described in the prompt $\mathcal{L}$, $[\Lambda, \mathcal{M}]$ along with the visual 3D keypoints of the live observation $\mathcal{K}_{live}$ are used as context for KAT that performs few-shot in-context imitation learning to generate a trajectory of hand joints actions, which is mapped to gripper poses and **executed** on the robot.

Processing the Long, Unlabelled Human Video

After recording the video of a human performing everyday tasks, we process it to remove unnecessary frames, i.e., frames that do not contain human hands. We achieve this automatically

by leveraging HaMeR [156] to detect the frames where hands are present. Consequently, we are left with one long, unlabelled video of smaller video clips, concatenated together, each containing only frames where a human interacts with various objects with their hands. This video is then passed to Gemini to proceed with the retrieval phase of R+X, as discussed in section 8.3. Fig. 8.7 shows a short segment of the uncut video we recorded. The frames marked with red correspond to frames where no human hands were detected. The frames where human hands were detected are marked with green. After processing the video with HaMeR [156] we remove the frames marked with red and are left with one long video of the green frames concatenated together.

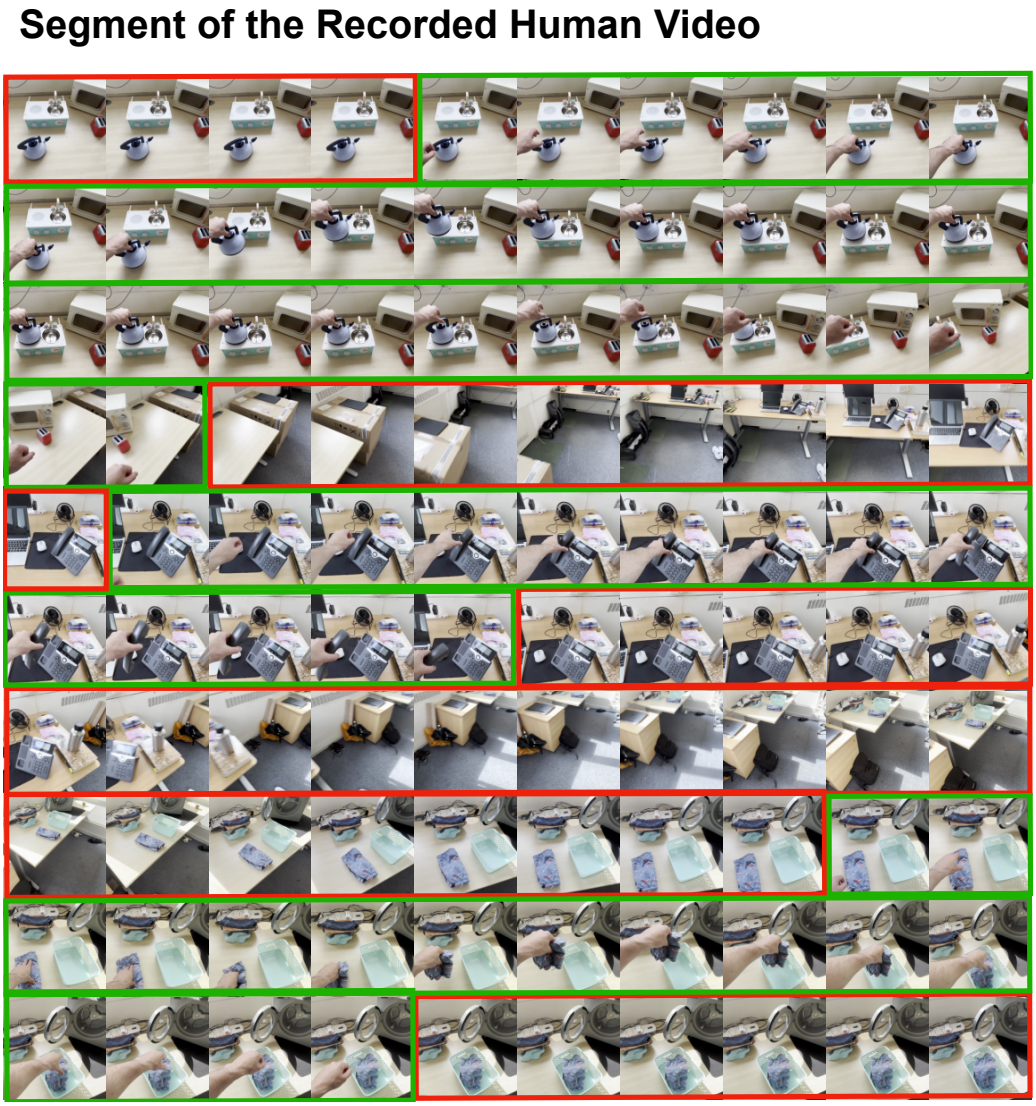


Figure 8.7: A short segment of our recorded long, unlabelled video of a human performing everyday tasks. The frames marked with red correspond to frames where no human hands were detected, while the frames marked with green correspond to frames where HaMeR [156] detected human hands. In practice, we discard the frames where no human hands are present and only retain a long video of frames concatenated together depicting only the human interacting with various objects.



Figure 8.8: An example of keypoints being tracked in a retrieved video clip. Notice how, thanks to the segmentation model [135], we can sample keypoints that stick to static parts of the scene, like the table.

Stabilisation of First Person Videos

As we record videos using a wearable camera, the point of view of the camera changes substantially frame by frame. However, our method is based on representing the visual inputs as fixed, 3D keypoints, and the actions as a series of 3D hand joints actions expressed in the same frame of the keypoints. Therefore, we need to compute the relative movement of the camera at each step in order to express everything in the original frame of reference, the first of the recorded video clip.

The computer vision literature has proposed several methods to address this problem [139, 145]. However, most of these techniques assume that the scene in front of the camera is fixed, and only the camera is moving. This is in sharp contrast with our setting, where the arms and hands of the user move in front of the camera, together with some of the objects. Classic techniques, in our experiments, failed to achieve a robust estimation of the movement of the camera, a task we hereby refer has the *stabilisation of the camera*. In the following section, we describe how we stabilise each video clip we retrieve from $\mathcal{H}$ while using R+X.

In order to tackle the aforementioned problem, we leverage the use of a series of recent computer vision models. The first one, CLIPSeg [135], is an open-vocabulary image segmentation model. The model receives an image and a series of language queries as an input, and outputs a segmentation mask localising the pixels of the desired objects, $\text{CLIPSeg}(\mathcal{O}_{live}, [\mathcal{L}_{obj,0}, \ldots, \mathcal{L}_{obj,n}]) \rightarrow M_{live} \in \mathbb{R}^{H \times W \times 1}$, where H and W are the height and width of the observation. We query CLIPSeg with some static parts of any scene, namely *"floor", "wall", "table"*. In addition, we query it with *"arm", "hand", "person"* in order to obtain segmentation masks of parts of the user arms, and remove it from the segmentation mask of the static objects.

Given the segmentation of such static objects, we sample a set of random 2D pixel keypoints in that area. We then track the movement of such keypoints frame by frame using TAPIR [49], a recent keypoint-tracking network. The position of these keypoints in 2D at each frame allows us to then project them in 3D using the depth-channel from the RGB-D camera we use. Given these sparse point-clouds, we can compute the relative $SE(3)$ rigid transformation between

each new frame and the first one. We perform this process separately for each video clip where human hands where detected as discussed in section 8.3.

This pipeline, in our experiments, has proven to be better than classic techniques [145], as it can handle the presence of the moving hands and arms of a person, together with the movement of some objects the user interacts with, as it learns to only focus on static parts of the environment, ignore the user, and track the selected keypoints over time.

Extracting Gripper Actions from Human Hands

To map the human hand joints to gripper actions we deploy different heuristics based on the type of interaction the human performs in the retrieved video clips. In scenarios where a robot is equipped with an end-effector with human-like fingers, such as a dexterous 5-fingered hand, such heuristics are not needed as the hand joints would map directly to the robotic, human-like fingers.

Fig. 8.9 shows instances of extracted hand joints using HaMeR [156] mapped to different gripper actions for grasping, pressing, and pushing tasks. For ease of illustration, this figure demonstrates how human hand joints are mapped to gripper poses from images captured in the human video. In practice, we deploy the following heuristics on the predictions made by each method. Our heuristics are applied to the Robotiq Gripper 2F85, but can be adapted accordingly to any other parallel jaw gripper.

Grasping. (Fig. 8.9, top) For tasks that require grasping we align spatially the tips of the gripper's fingers with the index and thumb tips of the hand. The vector joining the two tips defines an axis of rotation around which we rotate the gripper such that the bottom of the gripper is as close as possible to the middle point between the index mcp and the thumb dip. Aligning the gripper to these points enables us to obtain the gripper's pose. Consequently, for grasping tasks each method predicts, the index tip, thumb tip, index mcp and thumb dip joints. To compute the gripper close/open action, we employ an heuristic that measures the distance of the index and thumb tips and compares it to the robot gripper width.

Pressing. (Fig. 8.9, middle) For pressing tasks, we assume that the gripper is *closed*. To obtain the pose of the gripper we use three points at the point of contact of the two gripper fingers when the gripper is closed. Specifically, one point is at the tip of the fingers, the other in the middle, and the last at the bottom of the gripper's fingers. We align these three points to the index tip, index pip, index mcp, and index dip of the human hand respectively. Consequently, each method predicts the index tip, index pip, index mcp, and index dip.



Figure 8.9: Examples of different human hand interactions and extracted gripper actions based on three different heuristics for grasping, pressing, and pushing. The blue dots in the third column correspond to the 21 hand joints extracted using HaMeR [156]. Which heuristic to use is determined by Gemini at the Retrieval phase based on Gemini's understanding of the language command provided by the human. Note that for the pressing and pushing task the gripper is rendered as open, but in practice the gripper is closed as if it grasping something.

Pushing. (Fig. 8.9, bottom) Pushing tasks, are similar to pressing tasks, where we use the same three points on the gripper but to obtain the gripper's pose, we instead align these points with the joints on the hand halfway between the index and middle finger tip, pip, mcp and dip.

For all heuristics, we match the pose of the gripper with the joints on the hand using singular value decomposition.

Detecting which heuristic to use is done automatically by Gemini based on the human's language command during task execution. As Gemini exhibits strong semantic language understanding, it can determine whether the requested task from the human involves grasping, pressing, or pushing.

8.4 Experiments

Human Video. We collect the human video $\mathcal{H}$ using an Intel RealSense 455, worn by the user on their chest as shown in Figure 8.1. To reduce downstream computational time, we filter out each frame in which human hands are not visible right after recording. As our robot is single-armed, we limit ourselves to single hand tasks. However, our method could identically be applied to bimanual settings and dexterous manipulators. In our experiments, all videos are collected by the two operators, the co-first authors of the paper.

Robot Setup. At execution, we use a Sawyer robot with 7 degrees of freedom equipped with a RealSense 415 head-camera. The robot is equipped with a two-fingered parallel gripper, the Robotiq 2F-85. As the robot is not mobile, we setup different scenes in front of it with variations of the tasks recorded by the user, placing several different distractors for each task, while the human video was recorded in many different rooms. Although we have a wrist-camera mounted, we do not use that in our work. The robot is controlled using position control.

Tasks. To evaluate our proposed framework, we use a set of 12 everyday tasks, where the user interacts with a series of common household objects, listed in Fig. 8.6. We include movements like grasping, opening, inserting, pushing, pressing, and wiping.

- **Plate in Rack:** We randomise the position of the rack on the table in a 40cm by 30cm (width, height) area and its orientation in a $[-30, 30]$ degrees range. The robot starts with a plate grasped in its gripper. We place random distractors in half the runs, while half are distractors free. The task is successful if the plate can stand in one of the slots of the rack without falling over.

- **Push Rack in Dishwasher:** Due to the kinematics of the robot, we always put the dishwasher on the right side of the table, and randomise its position in a 35cm by 20cm area, and its orientation in a $[-20, 20]$ degrees range. As the object is considerably large, we cannot randomise excessively its starting point or it will end out of view/out of the robot's working area. We place random distractors in half the runs, while half are distractors free. The task is successful if the rack is no more than 5cm outside of the dishwasher, and starts being around 15cm outside of it.

- **Wipe Marker from Table:** We draw a random spot with a marker on an horizontally placed whiteboard, that emulates a mark on a table, some dirt, or something spilled. We randomise the shape of the spot, that is generally $15cm^2$ in area, and its position in a 40cm by 30cm area. The robot starts with a cleaning tool grasped. We place random

distractors in half the runs, while half are distractors free. The task is successful if more than 80% of the spot it erased and cleaned.

- **Grasp Beer**: We place a beer in a random spot on the table in a 40cm by 30cm area. In this case, we also place it 2 out of 10 times on top of other objects to test for hard spatial generalisation. We place random distractors in half the runs, while half are distractors free. The task is solved if the robot has grasped and lifted the beer.

- **Cloth in Washing Machine**: We place the washing machine on the right side of the table, randomising its position in a 20cm by 20cm area, and its orientation in a $[-20, 20]$ degrees range. We then place a random piece of clothing (generally clothing for kids/infants) on the left side of the table, and randomise its position in a 25cm by 25cm area, therefore testing for both spatial and object generalisation. We place random distractors in half the runs, while half are distractors free. The task is completed if the robot can place the cloth inside the hole of the washing machine, also if part is outside: as our toy washing machine is very small and its receptacle is shallow (around 15cm), it is hard also for a human to fit a cloth entirely in there.

- **Open Box:** We place one of two possible boxes of different sizes in a 30cm by 30cm area on the table, and randomise its orientation in a $[-20, 20]$ degrees range. We place random distractors in half the runs, while half are distractors free. The task is successful if, during the episode, the robot has partially opened the box. Opening it completely is often challenging due to kinematics constraints.

- **Kettle on Stove:** We place a toy stove on the table, randomising its position in a 30cm by 20cm area, and its orientation in a $[-20, 20]$ degrees range. We then randomly place a random kettle in a 20cm by 20cm area and randomise its orientation in a $[-30, 30]$ degrees range. By placing seen and unseen kettles, we test both for spatial and object generalisation. We place random distractors in half the runs, while half are distractors free. The task is successful if the kettle is on top of the stove, which we consider true if the intersection of the stove area and the kettle base is more than 50% than the stove area.

- **Close Microwave:** We place a toy microwave on the right side of the table, randomising its position in a 20cm by 20cm area and its orientation in a $[-20, 20]$ degrees range. Additionally, we open its door in a range between $[10, 45]$ degrees, We place random distractors in half the runs, while half are distractors free. We consider the task successful

if the microwave door is completely closed (it has a magnetic system that keeps it closed once it is completely pushed in position).

- **Cloth in Basket:** We place a basket on the right side of the table in a 20cm by 20cm area, and randomise its orientation in a $[-20, 20]$ degrees range. We place a random cloth (same as for the washing machine task) on the left side of the table, randomising its position in a 25cm by 25cm area.We place random distractors in half the runs, while half are distractors free. We consider the task successful if the cloth is contained in the basket, i.e. if lifting the basket, the cloth is lifted as well without falling on the table.

- **Pick Up Phone:** We randomly place one out of two possible phones on the table, randomising its position in a 30cm by 30cm area, and its orientation in a $[-30, 30]$ degrees range. By using different phones, one of which unseen at training, we also test for object generalisation. We place random distractors in half the runs, while half are distractors free. We consider the task successful if the robot has grasped and lifted the phone from its base.

- **Grasp Can:** Same setting as for Grasp Beer. In this case, we also use other unseen cans at test time to test for object generalisation.

- **Turn On Light:** We place a light bulb and a socket on the table, randomising their positions independently in a 30cm by 30cm area, and the socket orientation in a $[-20, 20]$ degrees range. We place random distractors in half the runs, while half are distractors free. The task is successful if the robot can press the light bulb's plug entirely in the socket, turning on the light.

Baselines. We compare R+X, and its retrieval and execution design, to training a single, language-conditioned policy. To obtain language captions from the human video, we use Gemini to autonomously caption snippets of the video, obtaining a *(observation, actions, language)* dataset. We finetune R3M (ResNet-50 version [87]) [147] and Octo [189] on this data. We extend R3M to also encode language via SentenceBERT and use a Diffusion Policy [35] head to predict actions from intermediate representations. We denote this version as R3M-DiffLang.
Octo One of baselines that we have compared against is a fine tuned version of Octo [189]. We have used the original code provided by the authors and adapted it for the specific input and output relevant to this work. More specifically, we designed the input of the model to be the RGB image of size 256×256 of the live observation, $\mathcal{O}_{live}$ and a language description of the task $\mathcal{L}$. The output, instead, is $\mathcal{J}_{live}$, the trajectory of desired hand joints 3D points to

Method / Task	Plate	Push	Wipe	Beer	Wash	Box	Kettle	Micro.	Basket	Phone	Can	Light	Avg.
R3M-DiffLang	0.5	0.7	0.4	0.7	0.5	0.5	0.4	0.8	0.7	0.4	0.7	0.3	0.55 [0.46, 0.64]
Octo	0.5	0.8	0.5	0.6	0.5	0.5	0.4	0.7	0.6	0.4	0.6	0.3	0.53 [0.44, 0.61]
R+X	**0.6**	0.8	**0.7**	**0.8**	**0.6**	**0.7**	**0.6**	0.8	0.7	**0.7**	**0.8**	**0.6**	**0.7** [0.61, 0.77]

Table 8.1: Result of the various methods on the 12 proposed tasks. We report success rates over 10 episodes per task, and also 95% confidence Wilson intervals over the aggregated average success rates.

be translated to gripper poses as described before. Unlike KAT, these methods require a fixed length of the output trajectory. We therefore pad it to 40 times steps. The output is ultimately a tensor of dimension 480, which corresponds to a flattened tensor of dimension $(40, 4, 3)$. The 4 predicted hand joints are described in section 8.3 along with how they are mapped to robot actions.

Having set the input and output dimensions, as well as modalities, the model has been finetuned starting from the provided checkpoint "octo-small-1.5" and it was trained until convergence.

R3M-DiffLang The other baseline we have compared against has been the combination of R3M [147] and [35], with the inclusion of language conditioning. The resulting model leveraged the ResNet-50 variant of R3M to encode the 256×256 live observation RGB image $\mathcal{O}_{live}$ and Sentence-BERT [167] to encode the language description of the task $\mathcal{L}$. The resulting feature vectors for the image and language description was then concatenated and passed to a UNet [170] trained via diffusion following the original code of [35]. The output of the diffusion head has the same size as the output of Octo, as we pad the trajectory $\mathcal{J}_{live}$ to 40 timesteps. In order to take advantage of the pre-trained vision knowledge of R3M, the encoder has been initialised with the provided weights and kept frozen, as it has been done for the language encoder. The diffusion policy head has been trained from scratch using our own dataset of trajectories until convergence was reached.

Results

Can R+X learn robot skills from long, unlabelled videos? How does it perform with respect to a monolithic language-conditioned policy? In these experiments, we evaluate the performance of R+X in learning a set of everyday skills. At deployment, we place the object to be interacted with in front of the robot, and issue a language command. We then run the retrieval and execution pipeline using the human video, the current observation and the language command. We run 10 episodes per task, randomising 1) the object pose 2) the

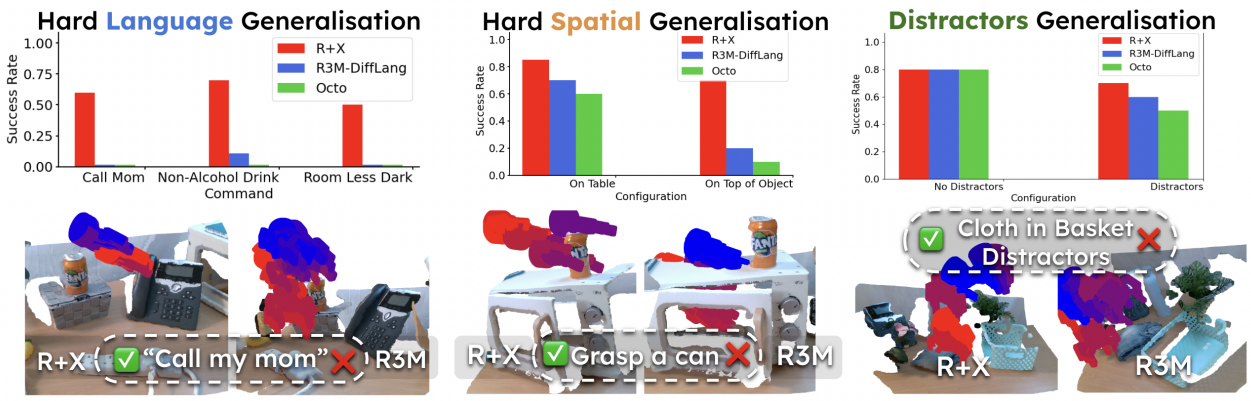


Figure 8.10: Examples of spatial, language and distractors generalisation experiments. Gripper trajectories move from red to blue. Plots provide success rates of each method on each generalisation experiment. Here we also report average success rates and 95% confidence Wilson intervals.
Hard Language Generalisation: R+X 0.60 [0.42, 0.75], **R3M** 0.03 [0.01, 0.17], **Octo** 0.00 [0.00, 0.11]
Hard Spatial Generalisation: **R+X** 0.75 [0.53, 0.89], **R3M** 0.45 [0.26, 0.66], **Octo** 0.35 [0.18, 0.57],
Distractors Generalisation R+X 0.75 [0.53, 0.89], **R3M** 0.70 [0.48, 0.85], **Octo** 0.65 [0.43, 0.82]

type and number of distractors 3) for tasks where it is possible, we swap the object to interact with, with another one from the same class to test for generalisation (e.g. a different can, a different piece of clothing, a different telephone).

In Table 8.1, we report the performance of R+X on these tasks, together with the baselines. As we demonstrate, the framework is able to tackle a wide range of everyday tasks, surpassing the monolithic policy baselines. These results prove the benefit of modeling language-conditioned learning from observation as distinct retrieval and execution phases, to fully leverage the abilities of recent Foundation Models.

What are the main sources of difference in performance between R+X and a monolithic policy? In these experiments, we investigate more in detail what changes in the inputs lead to the most noticeable difference in performance between R+X and the baselines, R3M-DiffLang and Octo. We explored two aspects, related to two properties of R+X:

Hard Spatial Generalisation: R+X, by retrieving videos of the desired task, can also extract a series of relevant keypoints $\mathcal{K}_{live}$ from the current observation $\mathcal{O}_{live}$, something not possible when using a single policy network. Going from RGB-D to a list of 3D points for inputs and outputs allows us to apply simple geometric data augmentation techniques for KAT, such as normalisation or random translations and rotations. This leads to a stronger spatial generalisation: in the *"grasp a can"* and *"grasp a beer"* tasks we test performance of

each method when the objects are on the table, or when they are positioned on top of other objects (a box, a microwave). By running 5 test episodes for each case, we demonstrate how R+X retains strong performance, while the performance of the baselines drop. Results and an example of the predicted actions can be seen in Fig. 8.10.

Hard Language Generalisation: By leveraging the language and video understanding abilities of recent large Vision Language Models, such as Gemini, R+X can interpret and execute nuanced commands. To evaluate this, we setup a scene with many objects, and ask the robot to perform three tasks: *"give me something to call my mom"*, *"give me a non-alcoholic drink"* and *"make the room less dark"*. We run 5 test episodes for each of these commands, modifying the position of the objects and the distractors. The language-conditioned policies struggle to interpret these commands, that are strongly out of distribution. R+X, on the other hand, leverages Gemini's ability to understand the meaning of these commands, as it is able to retrieve useful video clips from the human video (respectively, picking up the phone, grasping a Fanta can, and turning on the light). Results and an example of output gripper trajectories can be seen in Fig. 8.10.

Distractors Generalisation: One of the main differences between R+X and the baselines is that the former, after retrieving a set of video clips from $\mathcal{H}$ depicting the requested task, can extract a set of keypoints that are semantically and geometrically meaningful for the objects to interact with. These keypoints are generally robust to distractors, as they focus only on the DINO features that are common in all images, whereas distractors typically vary throughout the human video $\mathcal{H}$ and the execution. The baselines, being monolithic policies, receive as input the observation as it is, and must learn to generalise to the presence of distractors during the long training phase.

To evaluate the effect of distractors on the scene, we test R+X and the baselines on two tasks, "*grasp a can*" and "*put the cloth in the basket*", emulating the experimental scenario of the "*Hard Generalisations*" sections. We run 10 runs without distractors, and 10 with, explicitly measuring the performance in the two cases. We can see in the results of Fig. 8.10, how R+X is more robust to the presence of random distractor objects, highlighting the advantage of extracting semantically and geometrically meaningful keypoints *after* having received a language command and having retrieved the corresponding video clips.

Can R+X learn task sequentially over time? In these experiments, we demonstrate how R+X can learn tasks continually, with no need for any additional training or finetuning, while obtaining strong performance both on the new task and on the old ones, a desirable ability for a robot learning from an ever increasing dataset of human experience. To measure

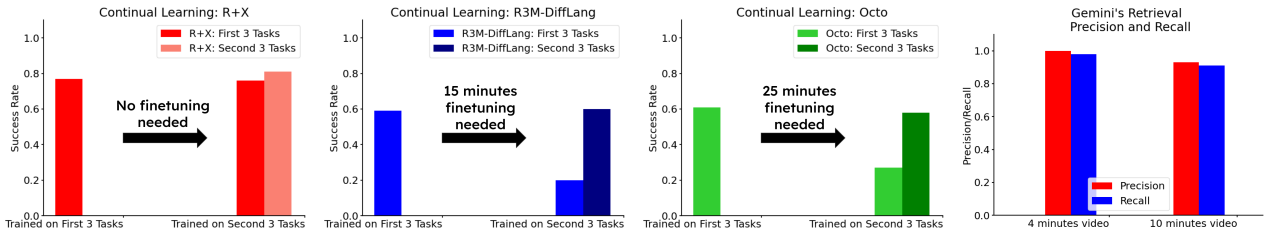


Figure 8.11: Left (barplots 1-3): We compare R+X and the baselines' ability to learn tasks in succession, and the time needed to learn such new tasks. Rightmost barplot: Gemini's retrieval performance.

this ability, and highlight the difference behaviour with respect to a single language-conditioned policy, we first collect 10 demos for 3 tasks. We train the baselines on these tasks (after extracting captions via Gemini as described for the experiments of Table 8.1) and evaluate them and R+X. Then, we add 10 demos of 3 new tasks, finetune the baselines, then measure performance on the 3 new tasks, and on the 3 old tasks for all methods. In Fig. 8.11, Left, we see how the performance of R3M-DiffLang and Octo deteriorates on the old task, due to the well known effect of catastrophic forgetting [205]. On the other hand, R+X performance does not deteriorate, due to the ability to retrieve from an ever-growing video of tasks and adapting the behaviour model at test time on the retrieved data.

If instead we train the baselines on all the data each time, the performance does not deteriorate: however, this leads to a substantial growth in time needed to train. As the dataset grows, those networks would need an increasing amount of time per each new added task in order to be retrained, while R+X does not need any training or finetuning.

How does the length of the videos affect the retrieval performance? In this experiments, we first collect a video of an operator providing 10 demos of 5 tasks (*kettle on stove, cloth in basket, pick up phone, cloth in washing machine, close microwave*). The length of the resulting video is 4 minutes. We then record a new video, recording how the operator provides 10 demos for the remaining tasks taken from 8.1, and concatenate to the original video. This new, longer video has a total length of 10 minutes. To empirically evaluate Gemini's ability at retrieving the correct video given a natural language instruction, we ask it to retrieve all the videos for each of the tasks recorded, and measure its precision and recall at retrieving those. By repeating the experiment on the 4 minutes and the 10 minutes videos, we can evaluate how the length of the video impacts its performance. In Fig. 8.11, Right, we show the mean over all tasks of retrieval's precision and recall. Results demonstrate how Gemini's performance is both very strong both in terms of precision and recall, but also how the same performance barely degrades when the length of the video more than doubles.

How does the performance change with the number of retrieved videos? R+X is



Figure 8.12: Examples of keypoints extracted for the same tasks, but with different views, settings, and target objects. Images are grouped in pairs column-wise. Keypoints are extracted between top and bottom images of each column.

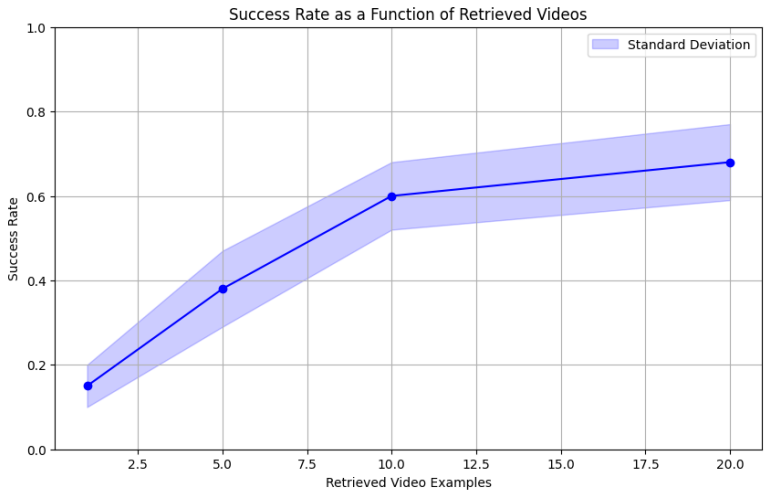


Figure 8.13: Success rate as a function of the amount of retrieved videos. We plot mean and standard deviation on 4 tasks, described below.

conditioned on a set of video clips depicting the requested tasks. How many videos clips need to be retrieved from the full human video to obtain strong performance? In this experiments, we manually limit the amount of videos that are retrieved for the "*pick up can, pick up telephone, cloth in basket and open box*" tasks, changing it between 1, 5, 10 and 20. We then run 10 test episodes for each of these values and compute the overall success rate. In Figure 8.13 we see how R+X can reach a strong performance also with as few as 10 retrieved videos.

8.5 Limitations

The method is currently bottlenecked by the performance of Vision Language Model, that has however seen a drastic improvement in recent time and will likely still increase. While we can learn a large series of everyday tasks, the errors arising from the hand pose prediction currently inhibits us from learning precise tasks. Furthermore, while the keypoints extraction leads to strong spatial generalisation properties, it can become a bottleneck in the presence of many similar objects: methods that find a dynamic number of keypoints might tackle this issue. Finally, due to the use of Foundation Models for retrieval and execution, our method has a few seconds delay between the issuing of the command and execution, that however has reduced dramatically over the last months with the release of faster models such as GPT-4o [151] or Gemini Flash [188].

8.6 Discussion

We presented R+X, a method to learn robot skills from long, unlabelled videos of users interacting with their environments, demonstrating its clear benefits over training monolithic language-conditioned policy networks. This method presents a fundamental paradigm shift from classic methods that rely on training a large model once on a fixed dataset, and instead builds on the in-context imitation learning methods presented in this thesis to design and deploy a method that can be conditioned on the spot on new human videos, and replicate the behaviour immediately. The capability for potential users to teach robots by simply wearing cameras in their natural environments can result in drastically better capabilities for robots and lower time needed on the user side to teach new skills.

Chapter 9

Conclusion

> The time is gone, the song is over
> Thought I'd something more to say.
>
> Time - Pink Floyd

This thesis has explored novel approaches to tackle one of the fundamental challenges in robot learning: the need for large datasets of demonstrations to teach robots new skills. While the recent success of foundation models in computer vision and natural language processing has shown how scaling up datasets and model sizes can lead to unprecedented capabilities, collecting similar amounts of robotics data remains prohibitively expensive and time-consuming. Throughout this work, we investigated two complementary approaches to address this challenge: first, designing algorithms that allow robots to efficiently collect their own data through self-supervised learning, and second, leveraging the semantic understanding and pattern recognition abilities of recent foundation models trained on abundant non-robotics data modalities.

The progression from Self-Replay (Chapter 4) through R+X (Chapter 8) demonstrates an increasingly powerful principle: robots do not need to rely only on robotics-specific data to learn manipulation skills. Self-Replay showed that robots can generate their own alignment training data; DINOBot showed that web-trained vision models provide sufficient geometric understanding for manipulation; KAT showed that web-trained language models can serve as imitation learning machines; and R+X showed that these ideas combine to enable learning from casual human videos. The practical implications are significant. A user wishing to teach a robot a new manipulation skill need not possess any robotics expertise, expensive teleoperation equipment, or spend tens to hundreds of hours teaching skills to robots. They simply wear a camera while performing the task in their natural environment, issue a language command, and the robot synthesises the appropriate behaviour. This dramatically lowers the barrier to

robot programming and opens possibilities for robots that continuously learn from the humans around them.

The **first part** of our research focused on enabling robots to autonomously collect training data that can then be used to train alignment policies in a self-supervised fashion. In Chapter 4, we introduced Self-Replay, a novel framework that allows a robot to learn multi-stage manipulation tasks from a single human demonstration. By decomposing complex tasks into coarse alignment and fine interaction phases, the robot can autonomously gather training data without constant human supervision or environment resets. We then extended this approach in Chapter 5 through the development of a retrieval-based system that enables generalisation to novel objects. This combination of retrieval, alignment, and replay proved to be remarkably efficient, requiring an order of magnitude less human demonstration time compared to traditional imitation learning approaches while achieving comparable or superior performance on a class of everyday tasks that can be decomposed into free-space alignment and last-inch interaction.

The **second part** of our research investigated how to leverage the recent breakthroughs in foundation models to reduce the need for robotics-specific data collection. In Chapter 6, we introduced DINOBot, demonstrating how the semantic and geometric understanding capabilities of vision foundation models can be effectively harnessed for robot manipulation without any additional training. We extend the methodologies explored in previous chapters, showing that the improvement of vision foundation models allows the robot to be deployed without any training time after receiving demos, relying entirely on the visual understanding of such models. Following on the same research direction, Chapter 7 introduced Keypoint Action Tokens (KAT), which showed how even large language models, despite being trained exclusively on text data, can serve as efficient few-shot imitation learning machines when provided with appropriate representations of visual observations and robot actions. These chapters show that, by training large models on enormous, unlabelled data of images and text from the web, we can bridge the gap between those data modalities and robotics. Albeit these datasets seem unrelated to robotic data, we demonstrate that the models that emerge from vast pre-training can indeed be instrumental for efficient robot manipulation learning.

These contributions collectively address the meta-solution challenge proposed in Chapter 1: rather than designing *specific solutions* for each manipulation task, our work demonstrates how to create a more general framework that can efficiently learn new behaviours. The self-supervised data collection methods presented in Chapters 4 and 5 dramatically reduce the human effort needed to teach robots new skills, while the foundation model approaches in

Chapters 6 and 7 showcase how robots can leverage knowledge from vast, non-robotics datasets. Chapter 8 builds on these developments to design a method that can teach robots in drastically more intuitive way, entirely from human videos. This represents a significant step toward more scalable robot learning, where the cost of teaching new skills no longer involves manually engineering techniques to let robots tackle such tasks in the physical world.

Looking ahead, this thesis suggests that the future of robot learning may lie in the combination of three elements. First, collecting expert demonstrations remains crucial for teaching robots complex manipulation skills that require human-level dexterity and decision-making. Second, enabling robots to autonomously gather their own data through self-supervised learning, in tasks where this is possible, can dramatically scale up data collection while minimising human effort. Third, leveraging and scaling foundation models trained on vast datasets from other domains can provide robots with robust semantic understanding and efficient adaptation capabilities. Although each approach has limitations when used in isolation, their combination could unlock unprecedented capabilities in robot learning. Future research should focus on developing frameworks that can seamlessly integrate these three elements, potentially leading to robots that can learn and adapt to the diverse challenges of human environments efficiently.

Bibliography

[1] Characters set. https://www.iana.org/assignments/character-sets/character-sets.xhtml. Accessed: 2024-09-14.

[2] Kinect for windows. https://learn.microsoft.com/en-us/windows/apps/design/devices/kinect-for-windows. Accessed: 2024-09-14.

[3] Polygon: All the money in the world couldn't make kinect happen. https://www.polygon.com/2020/1/14/21064608/microsoft-kinect-history-rise-and-fall. Accessed: 2024-09-14.

[4] Intel realsense. https://www.intelrealsense.com/. Accessed: 2024-09-14.

[5] Unicode. https://home.unicode.org/. Accessed: 2024-09-14.

[6] Daron Acemoglu and Simon Johnson. Learning from ricardo and thompson: Machinery and labor in the early industrial revolution, and in the age of ai. Working Paper 32416, National Bureau of Economic Research, May 2024. URL http://www.nber.org/papers/w32416.

[7] Daron Acemoglu and Pascual Restrepo. Automation and new tasks: How technology displaces and reinstates labor. *Journal of Economic Perspectives*, 33(2):3–30, May 2019. doi: 10.1257/jep.33.2.3. URL https://www.aeaweb.org/articles?id=10.1257/jep.33.2.3.

[8] K. Alton and M. van de Panne. Learning to steer on winding tracks using semi-parametric control policies. In *Proceedings of the 2005 IEEE International Conference on Robotics and Automation*, pages 4588–4593, 2005. doi: 10.1109/ROBOT.2005.1570827.

[9] Shir Amir, Yossi Gandelsman, Shai Bagon, and Tali Dekel. Deep vit features as dense visual descriptors, 2022. arXiv:2112.05814.

[10] Shir Amir et al. Deep vit features as dense visual descriptors. *arXiv:2112.05814*, 2021.

[11] Apple. Apple vision pro. URL https://www.apple.com/apple-vision-pro/.

[12] Kenjiro Ara. Labor, capital and land in economic growth. *Hitotsubashi Journal of Economics*, 2(1):26–41, 1961. ISSN 0018280X. URL http://www.jstor.org/stable/43295389.

[13] Max Argus, Lukas Hermann, Jon Long, and Thomas Brox. Flowcontrol: Optical flow based visual servoing. *arXiv preprint arXiv:2007.00291*, 2020.

[14] Robert U. Ayres and Benjamin Warr. *The Economic Growth Engine*. Number 13324 in Books. Edward Elgar Publishing, 2009. ISBN ARRAY(0x6c5d4848). URL https://ideas.repec.org/b/elg/eebook/13324.html.

[15] Jimmy Lei Ba. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

[16] Arpit Bahety, Priyanka Mandikal, Ben Abbatematteo, and Roberto Martín-Martín. Screwmimic: Bimanual imitation from human videos with screw space projection. *arXiv preprint arXiv:2405.03666*, 2024.

[17] Shikhar Bahl, Abhinav Gupta, and Deepak Pathak. Human-to-robot imitation in the wild. *arXiv preprint arXiv:2207.09450*, 2022.

[18] Shikhar Bahl, Russell Mendonca, Lili Chen, Unnat Jain, and Deepak Pathak. Affordances from human videos as a versatile representation for robotics, 2023. URL https://arxiv.org/abs/2304.08488.

[19] George A. Bekey and Kenneth Y. Goldberg. *Neural Networks in Robotics*. Kluwer Academic Publishers, USA, 1993. ISBN 079239268X.

[20] Homanga Bharadhwaj, Roozbeh Mottaghi, Abhinav Gupta, and Shubham Tulsiani. Track2act: Predicting point tracks from internet videos enables generalizable robot manipulation, 2024. URL https://arxiv.org/abs/2405.01527.

[21] Mariusz Bojarski, Davide Del Testa, Daniel Dworakowski, Bernhard Firner, Beat Flepp, Prasoon Goyal, Lawrence D Jackel, Mathew Monfort, Urs Muller, Jiakai Zhang, et al. End to end learning for self-driving cars. *arXiv preprint arXiv:1604.07316, year=2016*.

[22] Rishi Bommasani, Drew A Hudson, Ehsan Adeli, Russ Altman, Simran Arora, Sydney von Arx, Michael S Bernstein, Jeannette Bohg, Antoine Bosselut, Emma Brunskill, et al.

On the opportunities and risks of foundation models. *arXiv preprint arXiv:2108.07258*, 2021.

[23] Rinu Boney and Alexander Ilin. Semi-supervised and active few-shot learning with prototypical networks. *arXiv:1711.10856*, 2017.

[24] Sebastian Borgeaud, Arthur Mensch, Jordan Hoffmann, Trevor Cai, Eliza Rutherford, Katie Millican, George van den Driessche, Jean-Baptiste Lespiau, Bogdan Damoc, Aidan Clark, Diego de Las Casas, Aurelia Guy, Jacob Menick, Roman Ring, Tom Hennigan, Saffron Huang, Loren Maggiore, Chris Jones, Albin Cassirer, Andy Brock, Michela Paganini, Geoffrey Irving, Oriol Vinyals, Simon Osindero, Karen Simonyan, Jack W. Rae, Erich Elsen, and Laurent Sifre. Improving language models by retrieving from trillions of tokens, 2022.

[25] Jessica Borja-Diaz et al. Affordance learning from play for sample-efficient policy learning. *arXiv:2203.00352*, 2022.

[26] Dartnall HJ Bowmaker JK. Visual pigments of rods and cones in a human retina. *J Physiol.*, 01 1980.

[27] Angelo G. Bravos. Book review: Machine learning of robot assembly plans by arberto maria segre (kluwer academic publishers, 1988). *SIGART Bull.*, (109):17–18, jul 1989. ISSN 0163-5719. doi: 10.1145/70632.1059729. URL `https://doi.org/10.1145/70632.1059729`.

[28] Anthony Brohan et al. Rt-1: Robotics transformer for real-world control at scale. In *arXiv:2212.06817*, 2022.

[29] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

[30] Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging properties in self-supervised vision transformers, 2021. arXiv:2104.14294.

[31] Mathilde Caron et al. Emerging properties in self-supervised vision transformers. In *ICCV*, 2021.

[32] François Chaumette and Seth Hutchinson. Visual servo control. i. basic approaches. *Robotics Automation Magazine, IEEE*, 13:82 – 90, 01 2007. doi: 10.1109/MRA.2006.250573.

[33] Annie S. Chen, Suraj Nair, and Chelsea Finn. Learning generalizable robotic reward functions from "in-the-wild" human videos, 2021. URL `https://arxiv.org/abs/2103.16817`.

[34] Lili Chen, Kevin Lu, Aravind Rajeswaran, Kimin Lee, Aditya Grover, Michael Laskin, Pieter Abbeel, Aravind Srinivas, and Igor Mordatch. Decision transformer: Reinforcement learning via sequence modeling, 2021.

[35] Cheng Chi, Siyuan Feng, Yilun Du, Zhenjia Xu, Eric Cousineau, Benjamin Burchfiel, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion, 2023. arXiv:2303.04137.

[36] Krzysztof Choromanski, Valerii Likhosherstov, David Dohan, Xingyou Song, Andreea Gane, Tamas Sarlos, Peter Hawkins, Jared Davis, Afroz Mohiuddin, Lukasz Kaiser, David Belanger, Lucy Colwell, and Adrian Weller. Rethinking attention with performers, 2022. arXiv:2009.14794.

[37] Dan C Cireşan, Ueli Meier, Jonathan Masci, Luca M Gambardella, and Jürgen Schmidhuber. High-performance neural networks for visual object classification. *arXiv preprint arXiv:1102.0183*, 2011.

[38] Ignasi Clavera, Jonas Rothfuss, John Schulman, Yasuhiro Fujita, Tamim Asfour, and Pieter Abbeel. Model-based reinforcement learning via meta-policy optimization, 2018. arXiv:1809.05214.

[39] Open X-Embodiment Collaboration. Open x-embodiment: Robotic learning datasets and rt-x models, 2023. arXiv:2310.08864.

[40] Open X-Embodiment Collaboration. Open X-Embodiment: Robotic learning datasets and RT-X models. *arXiv:2310.08864*, 2023.

[41] John J Craig. *Introduction to robotics*. Pearson Educacion, 2006.

[42] Sudeep Dasari, Frederik Ebert, Stephen Tian, Suraj Nair, Bernadette Bucher, Karl Schmeckpeper, Siddharth Singh, Sergey Levine, and Chelsea Finn. Robonet: Large-scale multi-robot learning. *arXiv preprint arXiv:1910.11215*, 2019.

[43] Sudeep Dasari, Mohan Kumar Srirama, Unnat Jain, and Abhinav Gupta. An unbiased look at datasets for visuo-motor pre-training, 2023. arXiv:2310.09289.

[44] DeepSeek-AI, Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z. F. Wu, Zhibin Gou, Zhihong Shao, Zhuoshu Li, Ziyi Gao, Aixin Liu, Bing Xue, Bingxuan Wang, Bochao Wu, Bei Feng, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, Damai Dai, Deli Chen, Dongjie Ji, Erhang Li, Fangyun Lin, Fucong Dai, Fuli Luo, Guangbo Hao, Guanting Chen, Guowei Li, H. Zhang, Han Bao, Hanwei Xu, Haocheng Wang, Honghui Ding, Huajian Xin, Huazuo Gao, Hui Qu, Hui Li, Jianzhong Guo, Jiashi Li, Jiawei Wang, Jingchang Chen, Jingyang Yuan, Junjie Qiu, Junlong Li, J. L. Cai, Jiaqi Ni, Jian Liang, Jin Chen, Kai Dong, Kai Hu, Kaige Gao, Kang Guan, Kexin Huang, Kuai Yu, Lean Wang, Lecong Zhang, Liang Zhao, Litong Wang, Liyue Zhang, Lei Xu, Leyi Xia, Mingchuan Zhang, Minghua Zhang, Minghui Tang, Meng Li, Miaojun Wang, Mingming Li, Ning Tian, Panpan Huang, Peng Zhang, Qiancheng Wang, Qinyu Chen, Qiushi Du, Ruiqi Ge, Ruisong Zhang, Ruizhe Pan, Runji Wang, R. J. Chen, R. L. Jin, Ruyi Chen, Shanghao Lu, Shangyan Zhou, Shanhuang Chen, Shengfeng Ye, Shiyu Wang, Shuiping Yu, Shunfeng Zhou, Shuting Pan, S. S. Li, Shuang Zhou, Shaoqing Wu, Shengfeng Ye, Tao Yun, Tian Pei, Tianyu Sun, T. Wang, Wangding Zeng, Wanjia Zhao, Wen Liu, Wenfeng Liang, Wenjun Gao, Wenqin Yu, Wentao Zhang, W. L. Xiao, Wei An, Xiaodong Liu, Xiaohan Wang, Xiaokang Chen, Xiaotao Nie, Xin Cheng, Xin Liu, Xin Xie, Xingchao Liu, Xinyu Yang, Xinyuan Li, Xuecheng Su, Xuheng Lin, X. Q. Li, Xiangyue Jin, Xiaojin Shen, Xiaosha Chen, Xiaowen Sun, Xiaoxiang Wang, Xinnan Song, Xinyi Zhou, Xianzu Wang, Xinxia Shan, Y. K. Li, Y. Q. Wang, Y. X. Wei, Yang Zhang, Yanhong Xu, Yao Li, Yao Zhao, Yaofeng Sun, Yaohui Wang, Yi Yu, Yichao Zhang, Yifan Shi, Yiliang Xiong, Ying He, Yishi Piao, Yisong Wang, Yixuan Tan, Yiyang Ma, Yiyuan Liu, Yongqiang Guo, Yuan Ou, Yuduan Wang, Yue Gong, Yuheng Zou, Yujia He, Yunfan Xiong, Yuxiang Luo, Yuxiang You, Yuxuan Liu, Yuyang Zhou, Y. X. Zhu, Yanhong Xu, Yanping Huang, Yaohui Li, Yi Zheng, Yuchen Zhu, Yunxian Ma, Ying Tang, Yukun Zha, Yuting Yan, Z. Z. Ren, Zehui Ren, Zhangli Sha, Zhe Fu, Zhean Xu, Zhenda Xie, Zhengyan Zhang, Zhewen Hao, Zhicheng Ma, Zhigang Yan, Zhiyu Wu, Zihui Gu, Zijia Zhu, Zijun Liu, Zilin Li, Ziwei Xie, Ziyang Song, Zizheng Pan, Zhen Huang, Zhipeng Xu, Zhongyu Zhang, and Zhen Zhang. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning, 2025.

[45] Norman Di Palo and Edward Johns. Safari: Safe and active robot imitation learning with imagination. *arXiv preprint arXiv:2011.09586*, 2020.

[46] Norman Di Palo and Edward Johns. Learning multi-stage tasks with one demonstration via self-replay. In *CoRL*. PMLR, 2021.

[47] Norman Di Palo and Edward Johns. Keypoint Action Tokens Enable In-Context Imitation Learning in Robotics. *arXiv e-prints*, art. arXiv:2403.19578, March 2024. doi: 10.48550/arXiv.2403.19578.

[48] Norman Di Palo, Arunkumar Byravan, Leonard Hasenclever, Markus Wulfmeier, Nicolas Heess, and Martin Riedmiller. Towards a unified agent with foundation models. *arXiv preprint arXiv:2307.09668*, 2023.

[49] Carl Doersch, Yi Yang, Mel Vecerik, Dilara Gokay, Ankush Gupta, Yusuf Aytar, Joao Carreira, and Andrew Zisserman. Tapir: Tracking any point with per-frame initialization and temporal refinement. *arXiv preprint arXiv:2306.08637*, 2023.

[50] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale, 2021. arXiv:2010.11929.

[51] Alexey Dosovitskiy et al. An image is worth 16x16 words: Transformers for image recognition at scale. In *ICLR*, 2021.

[52] Danny Driess, Fei Xia, Mehdi S. M. Sajjadi, Corey Lynch, Aakanksha Chowdhery, Brian Ichter, Ayzaan Wahid, Jonathan Tompson, Quan Vuong, Tianhe Yu, Wenlong Huang, Yevgen Chebotar, Pierre Sermanet, Daniel Duckworth, Sergey Levine, Vincent Vanhoucke, Karol Hausman, Marc Toussaint, Klaus Greff, Andy Zeng, Igor Mordatch, and Pete Florence. Palm-e: An embodied multimodal language model, 2023. arXiv:2303.03378.

[53] Maximilian Du, Suraj Nair, Dorsa Sadigh, and Chelsea Finn. Behavior retrieval: Few-shot imitation learning by querying unlabeled datasets. *arXiv preprint arXiv:2304.08742*, 2023.

[54] Yan Duan, Marcin Andrychowicz, Bradly Stadie, OpenAI Jonathan Ho, Jonas Schneider, Ilya Sutskever, Pieter Abbeel, and Wojciech Zaremba. One-shot imitation learning. *Advances in neural information processing systems, 1087–1098*, 2017.

[55] Yan Duan, Marcin Andrychowicz, Bradly C. Stadie, Jonathan Ho, Jonas Schneider, Ilya Sutskever, Pieter Abbeel, and Wojciech Zaremba. One-shot imitation learning, 2017. arXiv:1703.07326.

[56] Anthony Brohan et al. Rt-2: Vision-language-action models transfer web knowledge to robotic control, 2023. arXiv:2307.15818.

[57] Gemini Team et al. Gemini: A family of highly capable multimodal models, 2023. arXiv:2312.11805.

[58] Hugo Touvron et al. Llama 2: Open foundation and fine-tuned chat models, 2023. arXiv:2307.09288.

[59] Kristen Grauman et al. Ego4d: Around the world in 3,000 hours of egocentric video, 2022. arXiv:2110.07058.

[60] Michael Ahn et al. Do as i can, not as i say: Grounding language in robotic affordances, 2022. arXiv:2204.01691.

[61] OpenAI et al. Gpt-4 technical report, 2023. arXiv:2303.08774.

[62] Rishi Bommasani et al. On the opportunities and risks of foundation models, 2022. arXiv:2108.07258.

[63] Haoshu Fang, Chenxi Wang, Minghao Gou, and Cewu Lu. Graspnet-1billion: A large-scale benchmark for general object grasping. *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 11441–11450, 2020. URL `https://api.semanticscholar.org/CorpusID:219964473`.

[64] Daniel J. Felleman and David C. van Essen. Distributed hierarchical processing in the primate cerebral cortex. *Cerebral cortex*, 1 1:1–47, 1991. URL `https://api.semanticscholar.org/CorpusID:9334496`.

[65] Chelsea Finn, Xin Yu Tan, Yan Duan, Trevor Darrell, Sergey Levine, and Pieter Abbeel. Deep spatial autoencoders for visuomotor learning. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 512–519. IEEE, 2016.

[66] Chelsea Finn, Tianhe Yu, Tianhao Zhang, Pieter Abbeel, and Sergey Levine. One-shot visual imitation learning via meta-learning. In *Conference on Robot Learning*, pages 357–368. PMLR, 2017.

[67] Peter Florence, Lucas Manuelli, and Russ Tedrake. Self-supervised correspondence in visuomotor policy learning. *IEEE Robotics and Automation Letters*, 5(2):492–499, 2019.

[68] Peter R Florence, Lucas Manuelli, and Russ Tedrake. Dense object nets: Learning dense visual object descriptors by and for robotic manipulation. *arXiv preprint arXiv:1806.08756*, 2018.

[69] James D. Foley and Andries Van Dam. *Fundamentals of interactive computer graphics*. Addison-Wesley Longman Publishing Co., Inc., USA, 1982. ISBN 0201144689.

[70] J. Fu, K. Luo, and S. Levine. Learning robust rewards with adversarial inverse reinforcement learning. *International Conference on Learning Representations*, 2018.

[71] Kunihiko Fukushima. Visual feature extraction by a multilayered network of analog threshold elements. *IEEE Transactions on Systems Science and Cybernetics*, 5(4):322–333, 1969. doi: 10.1109/TSSC.1969.300225.

[72] Kunihiko Fukushima. Neocognitron: A hierarchical neural network capable of visual pattern recognition. *Neural Networks*, 1(2):119–130, 1988. ISSN 0893-6080. doi: https://doi.org/10.1016/0893-6080(88)90014-7. URL `https://www.sciencedirect.com/science/article/pii/0893608088900147`.

[73] Gemini Team. Gemini: A Family of Highly Capable Multimodal Models. *arXiv e-prints*, art. arXiv:2312.11805, December 2023. doi: 10.48550/arXiv.2312.11805.

[74] Seyed Kamyar Seyed Ghasemipour, Richard Zemel, and Shixiang Gu. A divergence minimization perspective on imitation learning methods. *Conference on Robot Learning, 1259–1277*, 2019.

[75] Erin Gillam. An introduction to animal communication. *Nature Education*, 3(10), 2011.

[76] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *International Conference on Artificial Intelligence and Statistics*, 2010. URL `https://api.semanticscholar.org/CorpusID:5575601`.

[77] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

[78] Ian Goodfellow, Yoshua Bengio, Aaron Courville, and Yoshua Bengio. *Deep learning*, volume 1. MIT Press, 2016.

[79] Walter Goodwin, Ioannis Havoutis, and Ingmar Posner. You only look at one: Category-level object representations for pose estimation from a single example. In *6th Annual Conference on Robot Learning*, 2022. URL https://openreview.net/forum?id=lb7B5Rw7tjw.

[80] Alex Graves. Generating sequences with recurrent neural networks. *arXiv preprint arXiv:1308.0850*, 2013.

[81] Albert Gu and Tri Dao. Mamba: Linear-time sequence modeling with selective state spaces, 2023. arXiv:2312.00752.

[82] Abhishek Gupta, Vikash Kumar, Corey Lynch, Sergey Levine, and Karol Hausman. Relay policy learning: Solving long-horizon tasks via imitation and reinforcement learning. *arXiv preprint arXiv:1910.11956*, 2019.

[83] Denis Hadjivelichkov, Sicelukwanda Zwane, Lourdes Agapito, Marc Peter Deisenroth, and Dimitrios Kanoulas. One-shot transfer of affordance regions? affcorrs! In *Conference on Robot Learning*, pages 550–560. PMLR, 2023.

[84] M.T. Hagan and H.B. Demuth. Neural networks for control. In *Proceedings of the 1999 American Control Conference (Cat. No. 99CH36251)*, volume 3, pages 1642–1656 vol.3, 1999. doi: 10.1109/ACC.1999.786109.

[85] Siddhant Haldar, Jyothish Pari, Anant Rai, and Lerrel Pinto. Teach a robot to fish: Versatile imitation from one minute of demonstrations. *arXiv preprint arXiv:2303.01497*, 2023.

[86] R Hartley and A Zisserman. Multiple view geometry in computer. *Vision, 2nd ed., New York: Cambridge*, 2:5, 2003.

[87] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition, 2015.

[88] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[89] Kaiming He et al. Deep residual learning for image recognition. In *CVPR*, 2016.

[90] Dan Hendrycks and Kevin Gimpel. Gaussian error linear units (gelus). *arXiv preprint arXiv:1606.08415*, 2016.

[91] Nick Heppert, Max Argus, Tim Welschehold, Thomas Brox, and Abhinav Valada. Ditto: Demonstration imitation by trajectory transformation. *arXiv preprint arXiv:2403.15203*, 2024.

[92] J. Ho and S. Ermon. Generative adversarial imitation learning. *Advances in Neural Information Processing Systems, pages 4565–4573*, 2016.

[93] Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, Tom Hennigan, Eric Noland, Katie Millican, George van den Driessche, Bogdan Damoc, Aurelia Guy, Simon Osindero, Karen Simonyan, Erich Elsen, Jack W. Rae, Oriol Vinyals, and Laurent Sifre. Training compute-optimal large language models, 2022. arXiv:2203.15556.

[94] Jordan Hoffmann et al. Training Compute-Optimal Large Language Models. *arXiv e-prints*, art. arXiv:2203.15556, March 2022. doi: 10.48550/arXiv.2203.15556.

[95] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2(5):359–366, 1989. ISSN 0893-6080. doi: https://doi.org/10.1016/0893-6080(89)90020-8. URL `https://www.sciencedirect.com/science/article/pii/0893608089900208`.

[96] De-An Huang, Danfei Xu, Yuke Zhu, Animesh Garg, Silvio Savarese, Li Fei-Fei, and Juan Carlos Niebles. Continuous relaxation of symbolic planner for one-shot imitation learning. *arXiv preprint arXiv:1908.06769*, 2019.

[97] Wenlong Huang, Pieter Abbeel, Deepak Pathak, and Igor Mordatch. Language models as zero-shot planners: Extracting actionable knowledge for embodied agents, 2022. arXiv:2201.07207.

[98] Wenlong Huang, Chen Wang, Ruohan Zhang, Yunzhu Li, Jiajun Wu, and Li Fei-Fei. Voxposer: Composable 3d value maps for robotic manipulation with language models, 2023. arXiv:2307.05973.

[99] DAVID HUBEL and TORSTEN WIESEL. *Brain and Visual Perception: The Story of a 25-year Collaboration*. 03 2012. ISBN 9780195176186. doi: 10.1093/acprof:oso/9780195176186.001.0001.

[100] David H Hubel, Torsten N Wiesel, et al. Receptive fields of single neurones in the cat's striate cortex. *J physiol*, 148(3):574–591, 1959.

[101] Robert Hunt. The reproduction of colour. pages 1–702, 08 2005. doi: 10.1002/0470024275.

[102] Ahmed Hussein, Mohamed Medhat Gaber, Eyad Elyan, and Chrisina Jayne. Imitation learning: A survey of learning methods. *ACM Computing Surveys (CSUR), volume 50, number 2, 1–35*, 2017.

[103] Sergey Ioffe. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

[104] Alex Irpan, Chelsea Finn, Corey Harrison Lynch, Daniel Kappler, Eric Victor Jang, Frederik Ebert, Mohi Khansari, and Sergey Levine. Bc-z: Zero-shot task generalization with robotic imitation learning. 2021.

[105] Sergio Izquierdo, Max Argus, and Thomas Brox. Conditional visual servoing for multi-step tasks. In *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2190–2196, 2022. doi: 10.1109/IROS47612.2022.9981443.

[106] Vidhi Jain, Maria Attarian, Nikhil J Joshi, Ayzaan Wahid, Danny Driess, Quan Vuong, Pannag R Sanketi, Pierre Sermanet, Stefan Welker, Christine Chan, et al. Vid2robot: End-to-end video-conditioned policy learning with cross-attention transformers. *arXiv e-prints*, pages arXiv–2403, 2024.

[107] Stephen James, Michael Bloesch, and Andrew J Davison. Task-embedded control networks for few-shot imitation learning. *arXiv preprint arXiv:1810.03237*, 2018.

[108] Eric Jang, Sudheendra Vijayanarasimhan, Peter Pastor, Julian Ibarz, and Sergey Levine. End-to-end learning of semantic grasping. *arXiv preprint arXiv:1707.01932*, 2017.

[109] Eric Jang, Alex Irpan, Mohi Khansari, Daniel Kappler, Frederik Ebert, Corey Lynch, Sergey Levine, and Chelsea Finn. Bc-z: Zero-shot task generalization with robotic imitation learning, 2022.

[110] Edward Johns. Coarse-to-fine imitation learning: Robot manipulation from a single demonstration. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2021.

[111] Yuanchen Ju, Kaizhe Hu, Guowei Zhang, Gu Zhang, Mingrun Jiang, and Huazhe Xu. Robo-abc: Affordance generalization beyond categories via semantic correspondence for robot manipulation, 2024.

[112] Dmitry Kalashnikov et al. Scalable deep reinforcement learning for vision-based robotic manipulation. In *CoRL*, pages 651–673. PMLR, 2018.

[113] Mohi Khansari, Daniel Kappler, Jianlan Luo, Jeff Bingham, and Mrinal Kalakrishnan. Action image representation: Learning scalable deep grasping policies with zero real world data, 2020. arXiv:2005.06594.

[114] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv:1412.6980*, 2014.

[115] Günter Klambauer, Thomas Unterthiner, Andreas Mayr, and Sepp Hochreiter. Self-normalizing neural networks. *Advances in neural information processing systems*, 30, 2017.

[116] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C.J. Burges, L. Bottou, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 25. Curran Associates, Inc., 2012. URL `https://proceedings.neurips.cc/paper_files/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf`.

[117] Aviral Kumar, Aurick Zhou, George Tucker, and Sergey Levine. Conservative q-learning for offline reinforcement learning, 2020. URL `https://arxiv.org/abs/2006.04779`.

[118] Teyun Kwon, Norman Di Palo, and Edward Johns. Language models as zero-shot trajectory generators, 2023. arXiv:2310.11604.

[119] Michael Laskey, Jonathan Lee, Roy Fox, Anca Dragan, and Ken Goldberg. Dart: Noise injection for robust imitation learning. *arXiv preprint arXiv:1703.09327*, 2017.

[120] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proc. IEEE*, 86:2278–2324, 1998. URL `https://api.semanticscholar.org/CorpusID:14542261`.

[121] Michelle A Lee et al. Guided uncertainty-aware policy optimization: Combining learning and model-based strategies for sample-efficient policy learning. In *ICRA*. IEEE, 2020.

[122] A. Levas and M. Selfridge. A user-friendly high-level robot teaching system. 1:413–416, 1984. doi: 10.1109/ROBOT.1984.1087193.

[123] Sergey Levine, Chelsea Finn, Trevor Darrell, and Pieter Abbeel. End-to-end training of deep visuomotor policies. *The Journal of Machine Learning Research, volume 17, number 1, 1334-1373*, 2016.

[124] Sergey Levine, Aviral Kumar, George Tucker, and Justin Fu. Offline reinforcement learning: Tutorial, review, and perspectives on open problems, 2020. URL `https://arxiv.org/abs/2005.01643`.

[125] F. L. Lewis, A. Yesildirak, and Suresh Jagannathan. *Neural Network Control of Robot Manipulators and Nonlinear Systems*. Taylor & Francis, Inc., USA, 1998. ISBN 0748405968.

[126] Jacky Liang, Wenlong Huang, Fei Xia, Peng Xu, Karol Hausman, Brian Ichter, Pete Florence, and Andy Zeng. Code as policies: Language model programs for embodied control, 2023. arXiv:2209.07753.

[127] Seppo Linnainmaa. Taylor expansion of the accumulated rounding error. *BIT Numerical Mathematics*, 16:146–160, 1976. URL `https://api.semanticscholar.org/CorpusID:122357351`.

[128] David G Lowe. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2):91–110, 2004.

[129] Jiang Lu et al. Learning from very few samples: A survey. *arXiv:2009.02653*, 2020.

[130] Yao Lu et al. AW-opt: Learning robotic skills with imitation andreinforcement at scale. In *CoRL*, 2021.

[131] Corey Lynch and Pierre Sermanet. Grounding language in play.

[132] Corey Lynch, Mohi Khansari, Ted Xiao, Vikash Kumar, Jonathan Tompson, Sergey Levine, and Pierre Sermanet. Learning latent plans from play. *arXiv preprint arXiv:1903.01973*, 2019.

[133] Corey Lynch, Ayzaan Wahid, Jonathan Tompson, Tianli Ding, James Betker, Robert Baruch, Travis Armstrong, and Pete Florence. Interactive language: Talking to robots in real time, 2022.

[134] Corey Lynch et al. Learning latent plans from play. In *Conference on robot learning*. PMLR, 2020.

[135] Timo Lüddecke and Alexander S. Ecker. Image segmentation using text and image prompts, 2022.

[136] MagicLeap. Magicleap. URL `https://www.magicleap.com/`.

[137] Elman Mansimov and Kyunghyun Cho. Simple nearest neighbor policy method for continuous control tasks, 2018.

[138] Lucas Manuelli, Wei Gao, Peter Florence, and Russ Tedrake. kpam: Keypoint affordances for category-level robotic manipulation. In *Robotics Research: The 19th International Symposium ISRR*, pages 132–157. Springer, 2022.

[139] Hidenobu Matsuki, Riku Murai, Paul H. J. Kelly, and Andrew J. Davison. Gaussian splatting slam, 2024.

[140] Warren Mcculloch and Walter Pitts. A logical calculus of ideas immanent in nervous activity. *Bulletin of Mathematical Biophysics*, 5:127–147, 1943.

[141] Meta. Meta quest 3. URL `https://www.meta.com/gb/quest/quest-3/`.

[142] Suvir Mirchandani, Fei Xia, Pete Florence, Brian Ichter, Danny Driess, Montserrat Gonzalez Arenas, Kanishka Rao, Dorsa Sadigh, and Andy Zeng. Large language models as general pattern machines, 2023. arXiv:2307.04721.

[143] Suvir Mirchandani et al. Large Language Models as General Pattern Machines. *arXiv e-prints*, art. arXiv:2307.04721, July 2023. doi: 10.48550/arXiv.2307.04721.

[144] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning, 2013.

[145] Raul Mur-Artal, J. M. M. Montiel, and Juan D. Tardos. Orb-slam: A versatile and accurate monocular slam system. *IEEE Transactions on Robotics*, 31(5):1147–1163, October 2015. ISSN 1941-0468. doi: 10.1109/tro.2015.2463671. URL http://dx.doi.org/10.1109/TRO.2015.2463671.

[146] Ashvin Nair, Bob McGrew, Marcin Andrychowicz, Wojciech Zaremba, and Pieter Abbeel. Overcoming exploration in reinforcement learning with demonstrations. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6292–6299. IEEE, 2018.

[147] Suraj Nair et al. R3m: A universal visual representation for robot manipulation. *arXiv:2203.12601*, 2022.

[148] Donald A. Norman. *The design of everyday things*. Basic Books, [New York], 2002. ISBN 0465067107 9780465067107. URL http://www.amazon.de/The-Design-Everyday-Things-Norman/dp/0465067107/ref=wl_it_dp_o_pC_S_nC?ie=UTF8&colid=151193SNGKJT9&coliid=I262V9ZRW8HR2C.

[149] Catherine Olsson, Nelson Elhage, Neel Nanda, Nicholas Joseph, Nova DasSarma, Tom Henighan, Ben Mann, Amanda Askell, Yuntao Bai, Anna Chen, et al. In-context learning and induction heads. *arXiv prnote arXiv:2209.11895*, 2022.

[150] Shinji Oomori, Takeshi Nishida, and Shuichi Kurogi. Point cloud matching using singular value decomposition. *Artificial Life and Robotics*, 21:149–154, 06 2016. doi: 10.1007/s10015-016-0265-x.

[151] OpenAI. Gpt-4o. URL https://openai.com/index/hello-gpt-4o/.

[152] Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, et al. Dinov2: Learning robust visual features without supervision. *arXiv preprint arXiv:2304.07193*, 2023.

[153] Takayuki Osa, Joni Pajarinen, Gerhard Neumann, J Andrew Bagnell, Pieter Abbeel, and Jan Peters. An algorithmic perspective on imitation learning. *arXiv preprint arXiv:1811.06711*, 2018.

[154] Norman Di Palo and Edward Johns. On the effectiveness of retrieval, alignment, and

replay in manipulation. *IEEE Robotics and Automation Letters*, 9(3):2032–2039, 2024. doi: 10.1109/LRA.2024.3349832.

[155] Jyothish Pari et al. The surprising effectiveness of representation learning for visual imitation. *arXiv:2112.01511*, 2021.

[156] Georgios Pavlakos, Dandan Shan, Ilija Radosavovic, Angjoo Kanazawa, David Fouhey, and Jitendra Malik. Reconstructing hands in 3d with transformers, 2023.

[157] Frank L. Pedrotti, Leno M. Pedrotti, and Leno S. Pedrotti. *Introduction to Optics*. Cambridge University Press, 3 edition, 2017.

[158] Sören Pirk, Karol Hausman, Alexander Toshev, and Mohi Khansari. Modeling long-horizon tasks as sequential interaction landscapes. *arXiv preprint arXiv:2006.04843*, 2020.

[159] Vladimir N. Pokrovski. Energy in the theory of production. *Energy*, 28(8):769–788, 2003. ISSN 0360-5442. doi: https://doi.org/10.1016/S0360-5442(03)00031-8. URL `https://www.sciencedirect.com/science/article/pii/S0360544203000318`.

[160] Dean Pomerleau. Alvinn: An autonomous land vehicle in a neural network. In D.S. Touretzky, editor, *Proceedings of (NeurIPS) Neural Information Processing Systems*, pages 305 – 313. Morgan Kaufmann, December 1989.

[161] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners, 2019.

[162] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision, 2021. arXiv:2103.00020.

[163] Alec Radford et al. Learning transferable visual models from natural language supervision. In *ICML*, 2021.

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

[165] Rouhollah Rahmatizadeh, Pooya Abolghasemi, Ladislau Bölöni, and Sergey Levine. Vision-based multi-task manipulation for inexpensive robots using end-to-end learning

from demonstration. *2018 IEEE International Conference on Robotics and Automation (ICRA), 3758–3765, IEEE*, 2018.

[166] Meta Ray-Ban. Ray-ban smart glasses. URL `https://www.ray-ban.com/uk/ray-ban-meta-smart-glasses`.

[167] Nils Reimers and Iryna Gurevych. Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 11 2019. URL `https://arxiv.org/abs/1908.10084`.

[168] Martin Riedmiller, Roland Hafner, Thomas Lampe, Michael Neunert, Jonas Degrave, Tom Wiele, Vlad Mnih, Nicolas Heess, and Jost Tobias Springenberg. Learning by playing solving sparse reward tasks from scratch. In *International Conference on Machine Learning*, pages 4344–4353. PMLR, 2018.

[169] Javier Romero, Dimitrios Tzionas, and Michael J. Black. Embodied hands: modeling and capturing hands and bodies together. *ACM Transactions on Graphics*, 36(6):1–17, November 2017. ISSN 1557-7368. doi: 10.1145/3130800.3130883. URL `http://dx.doi.org/10.1145/3130800.3130883`.

[170] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation, 2015.

[171] Frank Rosenblatt. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 65 6:386–408, 1958. URL `https://api.semanticscholar.org/CorpusID:12781225`.

[172] Stéphane Ross, Geoffrey Gordon, and Drew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 627–635. JMLR Workshop and Conference Proceedings, 2011.

[173] Stéphane Ross, Geoffrey Gordon, and Drew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 627–635. JMLR Workshop and Conference Proceedings, 2011.

[174] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. Learning representations by back-propagating errors. *Nature*, 323:533–536, 1986. URL https://api.semanticscholar.org/CorpusID:205001834.

[175] Aseem Saxena, Harit Pandya, Gourav Kumar, Ayush Gaud, and K. Madhava Krishna. Exploring convolutional networks for end-to-end visual servoing, 2017. URL https://arxiv.org/abs/1706.03220.

[176] Juergen Schmidhuber. Annotated history of modern ai and deep learning. *arXiv preprint arXiv:2212.11279*, 2022.

[177] Younggyo Seo, Danijar Hafner, Hao Liu, Fangchen Liu, Stephen James, Kimin Lee, and Pieter Abbeel. Masked world models for visual control. In *Conference on Robot Learning*, pages 1332–1344. PMLR, 2023.

[178] Devavrat Shah and Qiaomin Xie. Q-learning with nearest neighbors. *NeurIPS*, 31, 2018.

[179] Rutav Shah and Vikash Kumar. Rrl: Resnet as representation for reinforcement learning. *arXiv preprint arXiv:2107.03380*, 2021.

[180] Lin Shao, Toki Migimatsu, Qiang Zhang, Karen Yang, and Jeannette Bohg. Concept2robot: Learning manipulation concepts from instructions and human demonstrations. *The International Journal of Robotics Research*, 40:1419 – 1434, 2020. URL https://api.semanticscholar.org/CorpusID:220069237.

[181] Dana Sharon et al. Synthesis of controllers for stylized planar bipedal walking. *ICRA*, pages 2387–2392, 2005.

[182] Mohit Shridhar, Lucas Manuelli, and Dieter Fox. Cliport: What and where pathways for robotic manipulation. In *Conference on Robot Learning*, pages 894–906. PMLR, 2022.

[183] Anthony Simeonov, Yilun Du, Yen-Chen Lin, Alberto Rodriguez Garcia, Leslie Pack Kaelbling, Tomás Lozano-Pérez, and Pulkit Agrawal. Se (3)-equivariant relational rearrangement with neural descriptor fields. In *Conference on Robot Learning*, pages 835–846. PMLR, 2023.

[184] Adam Smith. *The Wealth of Nations*. AuthorHouse, 2009. ISBN 9781420932065. URL http://books.google.ch/books?id=rBiqT86BGQEC.

[185] Olga Sorkine-Hornung and Michael Rabinovich. Least-squares rigid motion using svd. *Computing*, 1(1):1–5, 2017.

[186] R. Stresing, D. Lindenberger, and R. Kümmel. Cointegration of output, capital, labor, and energy. *The European Physical Journal B: Condensed Matter and Complex Systems*, 66(2):279–287, November 2008. doi: 10.1140/epjb/e2008-00412-. URL `https://ideas.repec.org/a/spr/eurphb/v66y2008i2p279-287.html`.

[187] Ilya Sutskever, James Martens, and Geoffrey Hinton. Generating text with recurrent neural networks. pages 1017–1024, 01 2011.

[188] Gemini Team. Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context, 2024.

[189] Octo Model Team, Dibya Ghosh, Homer Walke, Karl Pertsch, Kevin Black, Oier Mees, Sudeep Dasari, Joey Hejna, Tobias Kreiman, Charles Xu, Jianlan Luo, You Liang Tan, Lawrence Yunliang Chen, Pannag Sanketi, Quan Vuong, Ted Xiao, Dorsa Sadigh, Chelsea Finn, and Sergey Levine. Octo: An open-source generalist robot policy, 2024.

[190] Zachary Teed and Jia Deng. Raft: Recurrent all-pairs field transforms for optical flow. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part II 16*, pages 402–419. Springer, 2020.

[191] Eugene Valassakis, Norman Di Palo, and Edward Johns. Coarse-to-fine for sim-to-real: Sub-millimetre precision across the workspace. *arXiv preprint arXiv:2105.11283*, 2021.

[192] Eugene Valassakis et al. Demonstrate once, imitate immediately (dome): Learning visual servoing for one-shot imitation learning. *arXiv:2204.02863*, 2022.

[193] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2023. arXiv:1706.03762.

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

[195] Mel Vecerik, Jean-Baptiste Regli, Oleg Sushkov, David Barker, Rugile Pevceviciute, Thomas Rothörl, Raia Hadsell, Lourdes Agapito, and Jonathan Scholz. S3k: Self-supervised semantic keypoints for robotic manipulation via multi-view consistency. In *Conference on Robot Learning*, pages 449–460. PMLR, 2021.

[196] Mel Vecerik, Carl Doersch, Yi Yang, Todor Davchev, Yusuf Aytar, Guangyao Zhou, Raia Hadsell, Lourdes Agapito, and Jon Scholz. Robotap: Tracking arbitrary points for few-shot visual imitation. *arXiv preprint arXiv:2308.15975*, 2023.

[197] Sai Vemprala, Rogerio Bonatti, Arthur Bucker, and Ashish Kapoor. Chatgpt for robotics: Design principles and model abilities, 2023. arXiv:2306.17582.

[198] Pietro Vitiello, Kamil Dreczkowski, and Edward Johns. One-shot imitation learning: A pose estimation perspective. In *CoRL*, 2023.

[199] Johannes von Oswald, Eyvind Niklasson, Ettore Randazzo, João Sacramento, Alexander Mordvintsev, Andrey Zhmoginov, and Max Vladymyrov. Transformers learn in-context by gradient descent, 2023. arXiv:2212.07677.

[200] Vitalis Vosylius and Edward Johns. Few-shot in-context imitation learning via implicit graph alignment. In *CoRL*, 2023.

[201] Vitalis Vosylius and Edward Johns. Few-shot in-context imitation learning via implicit graph alignment, 2023. arXiv:2310.12238.

[202] Vitalis Vosylius and Edward Johns. Where to start? transferring simple skills to complex environments. In *CoRL*, pages 471–481. PMLR, 2023.

[203] Chen Wang, Linxi Fan, Jiankai Sun, Ruohan Zhang, Li Fei-Fei, Danfei Xu, Yuke Zhu, and Anima Anandkumar. Mimicplay: Long-horizon imitation learning by watching human play. *arXiv preprint arXiv:2302.12422*, 2023.

[204] Chen Wang, Haochen Shi, Weizhuo Wang, Ruohan Zhang, Li Fei-Fei, and C Karen Liu. Dexcap: Scalable and portable mocap data collection system for dexterous manipulation. *arXiv preprint arXiv:2403.07788*, 2024.

[205] Liyuan Wang, Xingxing Zhang, Hang Su, and Jun Zhu. A comprehensive survey of continual learning: Theory, method and application, 2024.

[206] Qianxu Wang, Haotong Zhang, Congyue Deng, Yang You, Hao Dong, Yixin Zhu, and Leonidas Guibas. Sparsedff: Sparse-view feature distillation for one-shot dexterous manipulation, 2023.

[207] Yixuan Wang, Zhuoran Li, Mingtong Zhang, Katherine Driggs-Campbell, Jiajun Wu, Li Fei-Fei, and Yunzhu Li. D^3fields: Dynamic 3d descriptor fields for zero-shot generalizable robotic manipulation, 2023.

[208] Yuxuan Wang, RJ Skerry-Ryan, Daisy Stanton, Yonghui Wu, Ron J Weiss, Navdeep Jaitly, Zongheng Yang, Ying Xiao, Zhifeng Chen, Samy Bengio, et al. Tacotron: Towards end-to-end speech synthesis. *arXiv preprint arXiv:1703.10135*, 2017.

[209] Bowen Wen et al. You only demonstrate once: Category-level manipulation from single visual demonstration. *arXiv:2201.12716*, 2022.

[210] Philipp Wu, Yide Shentu, Zhongke Yi, Xingyu Lin, and Pieter Abbeel. Gello: A general, low-cost, and intuitive teleoperation framework for robot manipulators, 2024. URL `https://arxiv.org/abs/2309.13037`.

[211] Danfei Xu, Suraj Nair, Yuke Zhu, Julian Gao, Animesh Garg, Li Fei-Fei, and Silvio Savarese. Neural task programming: Learning to generalize across hierarchical tasks. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3795–3802. IEEE, 2018.

[212] Tianhe Yu, Pieter Abbeel, Sergey Levine, and Chelsea Finn. One-shot hierarchical imitation learning of compound visuomotor tasks. *arXiv preprint arXiv:1810.11043*, 2018.

[213] Tianhe Yu, Chelsea Finn, Annie Xie, Sudeep Dasari, Tianhao Zhang, Pieter Abbeel, and Sergey Levine. One-shot imitation from observing humans via domain-adaptive meta-learning. *arXiv preprint arXiv:1802.01557*, 2018.

[214] Wenhao Yu, Nimrod Gileadi, Chuyuan Fu, Sean Kirmani, Kuang-Huei Lee, Montse Gonzalez Arenas, Hao-Tien Lewis Chiang, Tom Erez, Leonard Hasenclever, Jan Humplik, Brian Ichter, Ted Xiao, Peng Xu, Andy Zeng, Tingnan Zhang, Nicolas Heess, Dorsa Sadigh, Jie Tan, Yuval Tassa, and Fei Xia. Language to rewards for robotic skill synthesis, 2023. arXiv:2306.08647.

[215] Chengbo Yuan, Chuan Wen, Tong Zhang, and Yang Gao. General flow as foundation affordance for scalable robot learning, 2024. URL `https://arxiv.org/abs/2401.11439`.

[216] Andy Zeng, Pete Florence, Jonathan Tompson, Stefan Welker, Jonathan Chien, Maria Attarian, Travis Armstrong, Ivan Krasin, Dan Duong, Vikas Sindhwani, et al. Transporter networks: Rearranging the visual world for robotic manipulation. *arXiv preprint arXiv:2010.14406*, 2020.

[217] Albert Zhan, Philip Zhao, Lerrel Pinto, Pieter Abbeel, and Michael Laskin. A framework for efficient robotic manipulation. *arXiv preprint arXiv:2012.07975*, 2020.

[218] Tianhao Zhang, Zoe McCarthy, Owen Jow, Dennis Lee, Xi Chen, Ken Goldberg, and Pieter Abbeel. Deep imitation learning for complex manipulation tasks from virtual reality teleoperation. *IEEE International Conference on Robotics and Automation (ICRA), pages 1–8, 2018, IEEE*, 2018.

[219] Tianhao Zhang, Zoe McCarthy, Owen Jow, Dennis Lee, Xi Chen, Ken Goldberg, and Pieter Abbeel. Deep imitation learning for complex manipulation tasks from virtual reality teleoperation, 2018. URL `https://arxiv.org/abs/1710.04615`.

[220] Gaoyue Zhou, Victoria Dean, Mohan Kumar Srirama, Aravind Rajeswaran, Jyothish Pari, Kyle Hatch, Aryan Jain, Tianhe Yu, Pieter Abbeel, Lerrel Pinto, Chelsea Finn, and Abhinav Gupta. Train offline, test online: A real robot learning benchmark, 2023. arXiv:2306.00942.

[221] Yichen Zhu, Zhicai Ou, Xiaofeng Mou, and Jian Tang. Retrieval-augmented embodied agents, 2024.

[222] Yifeng Zhu, Arisrei Lim, Peter Stone, and Yuke Zhu. Vision-based manipulation from single human video with open-world object graphs. *arXiv preprint arXiv:2405.20321*, 2024.

[223] Hülya Kesici Çalışkan. Technological change and economic growth. *Procedia - Social and Behavioral Sciences*, 195:649–654, 2015. ISSN 1877-0428. doi: https://doi.org/10.1016/j.sbspro.2015.06.174. URL `https://www.sciencedirect.com/science/article/pii/S1877042815036538`. World Conference on Technology, Innovation and Entrepreneurship.