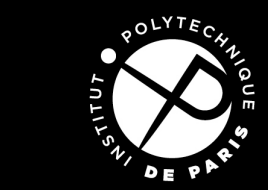
On the role of Actions and Machine Learning in Artificial Agent Perception.

Hugo Caselles-Dupré



INSTITUT POLYTECHNIQUE DE PARIS

NNT : 2021IPPAE006

Thèse de doctorat

On the role of Actions and Machine Learning in Artificial Agent Perception

Thèse de doctorat de l'Institut Polytechnique de Paris
préparée à École nationale supérieure de techniques avancées

École doctorale n°626 École Doctorale de l'Institut Polytechnique de Paris (ED IPP)
Spécialité de doctorat : Informatique, Données, IA

Thèse présentée et soutenue à Paris, le 10 Juin 2021, par

HUGO CASELLES-DUPRÉ

Composition du Jury :

Olivier Sigaud Professor, Sorbonne Université, Institut des Systèmes Intelligents et de Robotique	Président
Sylvain Argentieri Associate Professor, Sorbonne Université, Institut des Systèmes Intelligents et de Robotique	Rapporteur
Jean-Baptiste Mouret Director of Research, INRIA (équipe LARSEN)	Rapporteur
Irina Higgins Research Scientist, DeepMind	Examinateur
Alessandro Lazaric Research Scientist, Facebook Artificial Intelligence Research	Examinateur
David Filliat Professor, ENSTA Paris (U2IS)	Directeur de thèse
Michaël Garcia Ortiz Lecturer, City of London University	Co-directeur de thèse

Acknowledgements

Je suis heureux et nostalgique de conclure mon doctorat. Ces trois années furent une expérience humaine forte et enrichissante, qui, j'en suis conscient, influencera beaucoup ma personnalité et mes futures décisions. Je dois admettre que contrairement à beaucoup d'histoires que l'on raconte, ma thèse fut agréable et peu stressante. Je me suis toujours rendu compte que j'avais la chance de pouvoir prendre trois années pour me consacrer pleinement à une activité de recherche, avec très peu de contraintes intelectuelles, et un salaire correct. Cette chance là, je la dois à beaucoup de personnes, et j'aimerais profiter de cette section de mon thèse pour les remercier.

Premièrement j'aimerais remercier mes parents, François et Anne-Marie, et mon frère Paul pour m'avoir fourni une éducation qui m'a permis de développer curiosité, sens critique et ouverture d'esprit, trois atouts nécéssaires au travail de chercheur.

Deuxièmement j'aimerais remercier David et Michael, mes deux superviseurs, qui ont endossé leur rôle à merveille dans cette thèse. Ils m'ont grandement aidé à poursuivre ma thèse en étant toujours à l'écoute et sans jamais me mettre de pression inutile. Cela m'a permis de travailler en toute détente, dans des conditions idéales.

Je tenais à remercier mes collègues et collaborateurs de l'ENSTA, en particulier Timothée, René, Vyshakh et Antonin, qui m'ont donné un environnement de travail avec le juste équilibre entre calme et stimulation. Je remercie aussi mes collègues de Softbank, avec qui j'ai beaucoup échangé, notamment Alban, ce qui a contribué à l'aboutissement de mes projets de recherche. J'en profite aussi pour remercier mes collègues de l'INRIA Bordeaux, notamment Cédric et Adrien, avec qui j'ai passé d'excellents moments en conférence partout dans le monde.

Je voudrais aussi remercier mes amis (la boite, bt et le J, l'asint, ...) car c'est grâce à eux que la vie est sympathique et donne envie d'être vécue. Merci aussi pour toutes les surprises que vous m'avez fait pour la fin de ma thèse, ça m'a beaucoup touché.

Je me dois de faire une mention très spéciale à mes gavas d'Obvious, Gauthier et Pierre, sans qui cette thèse aurait sûrement été bien plus compliquée. Je suis heureux d'avoir trouvé un équilibre entre mes activités de recherche et nos activités d'artiste. Au delà d'un équilibre, ces deux activités

se sont révélées très complementaires: notre expérience ensemble est si forte qu'elle m'a permis de relativiser sur à peu près tout, et gagner un recul qui m'aide énormément dans la recherche, mais aussi dans la vie. Merci !

Finalement j'aimerais remercier Marine, mon bruhbruh, pour m'avoir accompagné durant ces trois années. C'est rigolo de se dire que la thèse nous effrayait un peu pour notre couple, alors que c'est passé comme une lettre à la poste. Mais c'est surement grâce à ta patience, ton attention et ton support quotidien. Je te remercie de m'avoir écouté lorsque je te racontais des choses concernant mes recherches qui ne t'interessait pas forcément, et aussi d'avoir partagé les moments de joies avec moi, comme lorsque j'ai appris pour le papier NeurIPS en vacances ensemble. Ces trois ans ne représentent qu'une fraction de notre histoire qui dure depuis beaucoup plus longtemps. J'espère qu'elle continuera beaucoup plus longtemps encore, et j'ai hâte de la vivre avec toi !

Contents

List of Figures

List of Tables

List of abbreviations

- ML Machine Learning
- NLP Natural Language Processing
- CL Continual Learning
- DL Deep Learning
- NN Neural Network
- CNN Convolutional Neural Network
- MLP Multi Layer Perceptron
- RL Reinforcement Learning
- SRL State Representation Learning
- SC Sensory Commutativity
- SCP Sensory Commutativity Probability
- SCOD Sensory Commutativity Object Detection
- SMCT SensoriMotor Contingencies Theory
- ALE Arcade Learning Environment
- DOF Degree Of Freedom
- GAN Generative Adversarial Networks
- VAE Variational AutoEncoder
- MDP Markov Decision Process
- EWC Elastic Weight Consolidation
- S-TRIGGER Self-Triggered Generative Replay
- PPO Proximal Policy Optimization

- SBDRL Symmetry-Based Disentangled Representation Learning
- AOD Active Object Detection

English summary

Automation is the medium by which the human species can free itself from the burden of tasks it has already solved. Such tasks are omnipresent in our daily lives, at home or in a professional context. A great quest in research is to build agents that can act and reason in the real-world, automating those solved tasks. For that, agents have to build a perception of their environments, just like humans do.

Directly programming those agents is infeasible because of the complexity of the world and its interaction. That is why learning-based approaches have been prevalent in research for the past 20 years. While supervised learning of algorithms using labeled data has provided numerous useful applications, having agents that perceive the world as well as biological agents do would require prohibitive amounts of labeled data. Research has thus opted for an unsupervised or weakly-supervised approach for building software algorithms that learn from data and can then be embedded in real robots that solve tasks in the real world. Most of the time those algorithms are trained in simulation for computational reasons (parallelization and higher speed than real-life).

On this concept, several learning approaches have been created for building agent perception. We have Machine Learning-based approaches that benefit from the Deep Learning revolution which allow the construction of hierarchical representation of data that can be used for task solving.

Among the Machine Learning approaches, we have several sub-fields that each tackle different aspects of perception that agents should have. State Representation Learning (SRL) focuses on learning representations of what the agents experience. SRL tries to mimic the ability of humans to summarize complex scenes into compositional objects and concepts. Continual Learning (CL) aims at solving the infamous catastrophic forgetting problem of neural networks, which forget everything they learned when presented with new data. Humans do not suffer from this problem as we have memory and selective forgetting mechanisms that allow us to continually learn throughout our lives. We finally have Reinforcement Learning (RL), which aims at learning to solve a task by maximizing the reward associated to it, a mechanism that is also present among biological agents.

On the other hand, we also have more original approaches that do

not necessarily have the same performances but are based on promising paradigms that could allow breakthroughs. Developmental robotics (DevRob) is a sub-field of robotics which aims at developing biological-inspired methods for learning on real robots. We also have what we shall call in this manuscript the Embodied Agent approaches, which are theoretical and practical considerations based on theories of perception developed in psychology. In these theories, the role of actions is crucial in the development of perception. We will use this as a basis for most of our contributions.

In this thesis we contribute to those sub-fields of research by developing theoretical insights and application algorithms which aim at creating agents with deeper levels of perception of their bodies and the environment. Specifically, we develop two novel approaches for Continual SRL and Continual RL with applications to real robots. We extend a theory on disentanglement for Representation Learning, by showing the crucial role of actions in learning. We finally propose a novel learning mechanism for embodied agents based on the sensory commutativity of action sequences: we take inspiration from EA theories and develop theoretical insights as well as learning algorithms for object detection and self-body discovery.

To conclude, in this thesis we hope to have shed light on promising ways of developing agent perception using learning mechanisms.

Résumé en français

L'automatisation est le moyen par lequel l'espèce humaine peut se libérer du fardeau des tâches qu'elle a déjà résolues. Ces tâches sont omniprésentes dans notre vie quotidienne, à la maison ou dans un contexte professionnel. Une grande ambition dans la recherche est de créer des agents capables d'agir et de raisonner dans le monde réel, en automatisant des tâches résolues. Pour cela, nous supposons que les agents doivent construire une perception de leur environnement, tout comme les humains.

La programmation directe de ces agents est impossible en raison de la complexité du monde et de ses interactions. C'est pourquoi les approches fondées sur l'apprentissage ont prévalu dans la recherche au cours des 20 dernières années. Alors que l'apprentissage supervisé des algorithmes utilisant des données étiquetées a fourni de nombreuses applications utiles, avoir des agents qui perçoivent le monde aussi bien que des agents biologiques exigerait des quantités prohibitives de données étiquetées. La recherche a ainsi opté pour des approches non supervisées ou faiblement supervisées pour construire des algorithmes qui apprennent à partir des données et peuvent ensuite être intégrés dans de vrais robots qui résolvent des tâches dans le monde réel. La plupart du temps, ces algorithmes sont entraînés en simulation pour des raisons de coûts de calcul (parallélisation et vitesse plus élevée que dans la vie réelle).

Sur ce concept, plusieurs approches d'apprentissage ont été créées pour la perception des agents de construction. Nous avons des approches basées sur le Machine Learning qui bénéficient de la révolution Deep Learning qui permettent la construction d'une représentation hiérarchique des données pouvant être utilisée pour la résolution de tâches.

Parmi les approches d'apprentissage automatique, nous avons plusieurs sous-domaines qui abordent chacun différents aspects de la perception que les agents devraient avoir. L'apprentissage de répresentations d'états (ARE) se concentre sur l'apprentissage des représentations de l'expérience des agents. Le ARE essaie d'imiter la capacité des humains à résumer des scènes complexes en objets et concepts de composition. L'apprentissage continu vise à résoudre le célèbre problème d'oubli catastrophique des réseaux de neurones, qui oublient tout ce qu'ils ont appris lorsque de nouvelles données leur sont présentées. Les humains ne souffrent pas de ce problème car nous avons

une mémoire et des mécanismes d'oubli sélectifs qui permettent d'apprendre continuellement tout au long de notre vie. Nous avons enfin l'apprentissage par renforcement, qui vise à apprendre à résoudre une tâche en maximisant la récompense qui lui est associée, mécanisme qui est également présent chez les agents biologiques.

D'un autre côté, nous avons aussi des approches plus originales qui n'ont pas forcément les mêmes performances mais reposent sur des paradigmes prometteurs qui pourraient permettre des progrès de recherche. La robotique développementale est un sous-domaine de la robotique qui vise à développer des méthodes d'apprentissage inspirées de la biologie sur de vrais robots. Nous avons aussi ce que nous appelons dans ce manuscrit les approches de l'agent incarné, qui sont des considérations théoriques et pratiques basées sur les théories de la perception développées en psychologie, philosophie et sciences cognitives. Dans ces théories, le rôle des actions est crucial dans le développement de la perception. Nous utiliserons cela comme base pour la plupart de nos contributions.

Dans cette thèse, nous contribuons à ces sous-domaines de recherche en développant des connaissances théoriques et des algorithmes d'application qui visent à créer des agents avec des niveaux plus profonds de perception de leur corps et de l'environnement. Plus précisément, nous développons deux approches novatrices pour l'apprentissage de représentation d'états et l'apprentissage continu avec des applications à de vrais robots. Nous étendons une théorie sur la désintrication pour l'apprentissage de représentation, en montrant le rôle crucial des actions dans l'apprentissage. Nous proposons enfin un nouveau mécanisme d'apprentissage des agents incarnés basé sur la commutativité sensorielle des séquences d'action: nous nous inspirons des théories sur la perception et développons des connaissances théoriques ainsi que des algorithmes d'apprentissage pour la détection d'objets et la découverte du corps.

Pour conclure, dans cette thèse, nous espérons avoir mis en lumière des moyens prometteurs de développer la perception des agents à l'aide de mécanismes d'apprentissage.

Chapter 1

Introduction

Contents

1.1 The role of Automation

Humanity benefits greatly from automation. Automation is the medium by which the human species can free itself from the burden of tasks it has

already solved. By solved we mean that given the current knowledge, achieving the task is doable within a reasonable budget.

1.1.1 Historical perspective

Historically, humans have automated a large number of tasks, and by this means they have improved their lives. For instance in agriculture: first, the wheel allowed us to transport heavy objects. Then came more elaborate machines that allowed us to automate the watering of fields, and finally today we have complex machines that automate the whole process of agriculture almost totally. Thanks to this automation, the human species has more food at its disposal, which allows it to grow in other domains without having to focus on the burden of agriculture.

Tasks that are automatable are omnipresent in our daily lives, at home or in a professional context. At home, all chores are repeated every so often. At work, a large class of actual actions in the workflow require a low intellectual effort, while being essential for accomplishing greater goals. These tasks are often considered a burden for the person. While this is a complex question when considering all the social and economical context, it is logical to believe that such tasks should be automated. As a disclaimer, we do not make any political, social or economical considerations in this purely scientific work. Those considerations are crucial for the application of automation mechanisms such as the one discussed in this thesis.

Think of it this way: most of the processes we know in our lives would radically change if a multi-purpose learning robot was available at a reasonable price. Only a restricted amount of activities would not be affected. Those which require high-level intellectual effort and are non-plannable in advance: creativity, strategy, brainstorming. The present work does not consider automating those tasks. We consider agents able to solve repetitive and unpleasant tasks such as domestic chores or cashier work.

This question of how to create a multi-purpose robot has been studied from various points of view during the last 50 years. Softbank for instance has invested in several robotics startups including Softbank Robotics Europe (SBRE, formerly known as Aldebaran, the company financing this thesis in the CIFRE framework) or Boston Dynamics. This PhD is partly supported by industry (SBRE) and academia (ENSTA Paris - INRIA Flowers), which shows that both side of research have interest in the field. SBRE produces the robot Pepper, which is a robot that allows navigation and manipulation in human environments. However, the robot is far from being able to be autonomous. We now ask: how can such a robot be created? Accordingly, in the thesis, we will focus on the algorithms, i.e. the software, and will not discuss the making of robots, i.e. the hardware.

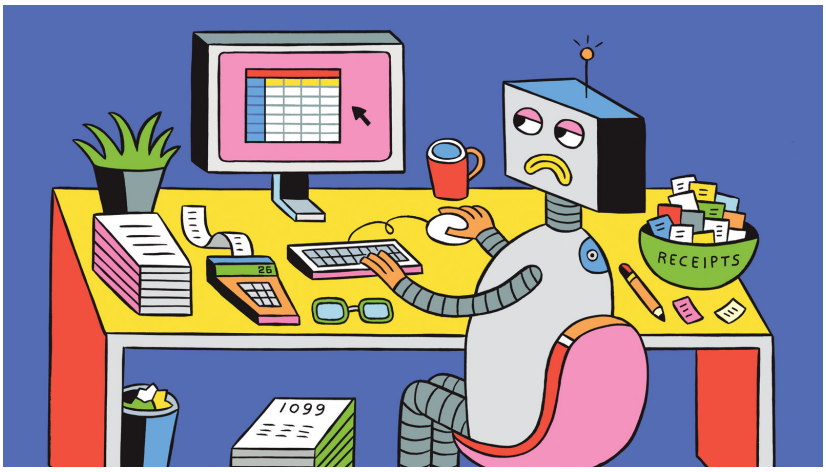


Figure 1.1: Automation: having machines do unpleasant work instead of us.

1.1.2 Robots in Human Environments

In a factory, the environment is precisely controlled, all actions can be predicted perfectly and there are no surprise items for the robot to interact with. Moreover, these environments never change so that a pre-programmed robot can continue performing well for extended periods of time. Otherwise, if the task changes, the robots need reprogramming. Hence, the robots do not need to have very detailed perception of their environment.

We are interested in having robots act in human environments. Contrary to a clean warehouse, or a robotic line in a factory, the environments humans use are complex and designed for human perception. Those environments are cluttered, dynamic and ever-changing. The number of objects to interact with is high, and objects vary in sizes, shapes, textures and colors. The background always changes, the weather changes daily and the season changes over a longer period of time, etc. Everything considered, the number of factors influencing perception and action is enormous.

A robot in a human environment thus needs to understand very complex scenes to plan for actions that fulfill tasks. It should know how to adapt to the changes that are continually applied to the environment. As a solution, we could equip the environments with cameras and motion sensors, and perceive the full scene not from the point of view of the robot. However, in our research, and for easier deployment in general, we want an agent to be able to act and learn in any new environment, without depending on equipping this environment further. This type of robots should have first person sensors, to avoid the need of preparing the environment everywhere we want to have a robot, and enough degrees of freedom to allow manipulation and navigation. We end up with a complex task: how can such a robot perceive these human environments?

1.2 The problem of perception

Perception is the medium by which agents organize and interpret sensory stimuli, in order to reason and act in an environment using their available actions [73]. Although perception is often reduced to a data processing problem in Computer Science, where perception (images) and action (motor commands) are separate standalone problems, many theories of perception insist on the role of action in the problem of perception. Perception is about learning how sensations change given actions.

1.2.1 Hypothesis

As an hypothesis, we are considering agents which we call embodied: they are situated in an environment, equipped with a body that allows movement, navigation and manipulation. Another hypothesis we state is that agents should build a perception of their environments close to ours, to easily learn to act in them. That is to say if an agent perceives the world as we do, in a compositional manner and in high-level terms (concepts of objects, scenes, and not pixel-level information), it will be much easier for it to solve the required tasks. This leads us naturally to the question of visual perception for embodied agents, and how to develop it using learning algorithms.

To have a perception close to a biological agent, embodied agents should for example: recognize the different elements and agents in the world (other agents, objects and their properties, the environment, etc), be aware of space and time and how physics work, be capable of navigation and manipulation in the environment and have a conceptual understanding of the relation between cause and effects. This list is not exhaustive but gives a sense of how complex the problem of perception is. In this thesis, we tackle sub-problems like object detection, representation learning or continual learning and aim at making progress on one particular question at a time.

1.2.2 The role of action in perception

More specifically, in this thesis, we are interested in the role of action in the emergence of perception. Theories of perception like Sensorimotor Contingencies Theory [123] or Gibson's visual perception theory [53] put forward the crucial role of actions in the construction of perception. In our contributions, we highlight this role by showing that invariants can emerge from the combination of observations and motor commands data, i.e. the sensorimotor stream of data, that agents experience. From those invariant we can deduce learning mechanisms that can be applied using Machine Learning models.

1.2.3 Learning instead of programming

Directly programming robots to act in the real world is infeasible because of the complexity of the world and its interaction. Indeed, the world is very diverse, and one person's bedroom will typically not be visually similar to another person's bedroom, although it will most likely contain the same element: a bed, a desk, a wardrobe, etc. It is thus impossible to directly program an agent to tidy a bedroom using RGB-images without having access to a mechanism that understands the high-level aspect of the visual scene. That is why learning-based approaches have been prevalent in this research field for the past 20 years.

While supervised learning of algorithms using labeled data has provided numerous useful applications, having agents that perceive the world as biological agents do would require prohibitive amounts of labeled data. It seems unfeasible to label enough data to make an agent learn everything about the world. Research has thus opted for an unsupervised or weakly-supervised approach for building algorithms that learn from data and can then be embedded in real robots that solve tasks in the real world. Moreover, as highlighted above, perception learning can benefit from actions, which can't be exploited in a static dataset, thus requiring learning directly on the real robot. However, most of the time those algorithms are trained in simulation for computational reasons (parallelization and higher speed than real-life).

We will therefore describe current approaches for building algorithms that help artificial agents understand their surroundings, and propose new approaches for doing so, that we will test in simulation.

1.3 Research directions for the problem of perception

Various approaches have been investigated along the two research directions we have highlighted. The work presented in this thesis will develop along those two lines following particular approaches that are described thereafter.

1.3.1 Machine Learning approaches

The Deep Learning revolution which started in the 2010s has led to a plethora of novel approaches in the Machine Learning field to the problem of perception and common sense. With the hope brought by the astounding results in image recognition, text-to-speech, automatic translation and several other fields, a large part of research lab around the world has focused on developing these approaches, eventually becoming the standard approach to any AI-related problem. This naturally also applies to the problem of learning perception for agents. We focus on three current standard approaches.

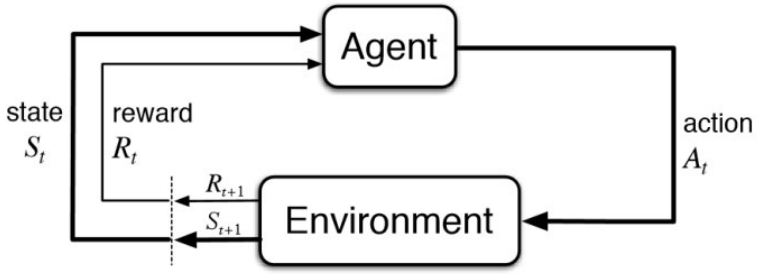


Figure 1.2: The Reinforcement Learning framework.

Reinforcement Learning

Reinforcement learning (RL) is an area of machine learning concerned with how software agents ought to take actions in an environment in order to maximize the notion of cumulative reward [113, 161]. The paradigm is illustrated in Fig.1.2. Reinforcement learning is one of three basic machine learning paradigms, alongside supervised learning and unsupervised learning. Reinforcement learning differs from supervised learning in not needing labelled input/output pairs be presented. Instead the focus is on finding a balance between exploration (of uncharted territory) and exploitation (of current knowledge). The environment is typically stated in the form of a Markov decision process (MDP).

RL is interesting in our work because with Deep Reinforcement Learning, it is now possible to apply RL to large state spaces such as images, and therefore learn to interpret these sensations while learning to solve a task, thus strongly relating perception to action.

State Representation Learning

Representation learning algorithms are designed to learn abstract features that characterize data. State representation learning (SRL) [96, 136] focuses on a particular kind of representation learning where learned features are in low dimension, evolve through time, and are influenced by actions of an agent. The paradigm is illustrated in Fig.1.3. The representation is learned to capture the essence and variations in the environment generated by the agent's actions; this kind of representation is particularly suitable for robotics and control scenarios. In particular, the low dimension characteristic of the representation helps to overcome the curse of dimensionality, provides easier interpretation and utilization by humans and can help improve performance and speed in policy learning algorithms such as reinforcement learning.

Disentanglement is also a property of SRL that is investigated in the problem of building perception using SRL. It's an unsupervised learning

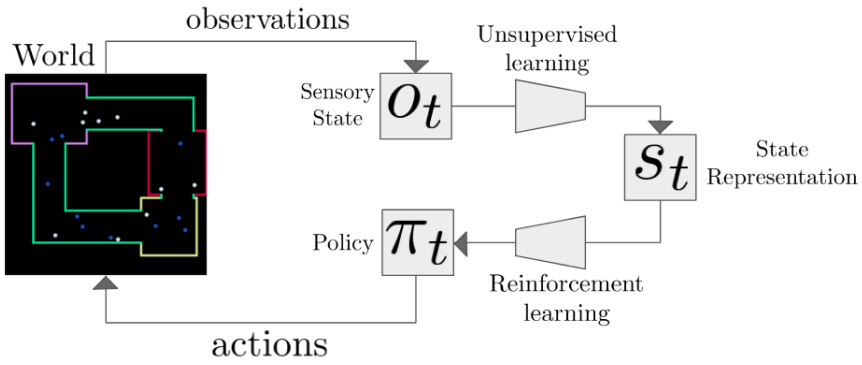


Figure 1.3: The State Representation Learning paradigm is decoupled in two phases: first learn a representation model for collected sensory states o_t, then feed the state representations s_t as input to a policy $\pi_t = \pi(s_t)$, which is optimized to solve a task using RL for example. Previous work [136] has shown this can be more efficient than using directly sensory states for RL.

technique that breaks down, or disentangles, each feature into narrowly defined variables and encodes them as separate dimensions. The reasoning behind this approach is to make it easier to extract useful information when building classifiers or other predictors. Several definitions of this intuitive concept have been proposed [104, 68, 10].

Typically, SRL states (with or without disentanglement techniques) are learned and then fed to RL algorithms, as presented in Fig.1.3. Instead of directly learning on the raw observations that are often high-dimensional and complex, it's easier for the RL algorithms to learn efficient policies when it only has to act on a restricted dimension space that corresponds to the factors of variation of the data that the agent receives, which SRL aims at building.

Continual Learning

The long term goal of creating agents capable of learning during extended periods of time, without constant supervision, will require the capability of continually learning about their environment. Indeed, learning about the world is inherently a progressive/incremental task, since the data distribution experienced by the agent can change both spatially (e.g. countries) and temporally (e.g. seasons). Classic ML approaches are not designed to handle this case, where the data distribution evolves through time. The field of Continual Learning has emerged to alleviate these problems and create algorithms that can learn continually as data evolves [125, 97].

In the case of the perception problem, this can involve learning visual features from an agent's surroundings, and continually updating these fea-

tures as its life progresses and the environment evolves, without forgetting the important features of the past.

1.3.2 Embodied agent approaches and developmental robotics

Aside from Machine Learning approaches presented above, there exists other approaches that tackle the problem of perception. These approaches are notably less popular, because they do not benefit from the exceptional performances and demonstrations that deep learning has. This does not mean that the ideas developed are not original, interesting, and worth pursuing. In this thesis, we will present such promising ways to progress on the problem of perception.

Embodied perception theories

In order to build agents that perceive the world as we do, we can take inspiration from theories of perception developed in the field of psychology and neuroscience. In this manuscript, we particularly focus on two approaches: Sensorimotor contingencies theory (SMCT) and Gibson's theory of visual perception. These approaches aim at understanding how perception emerges in embodied agents placed in the world.

Sensorimotor contingencies theory (SMCT) [123] is a theory of visual perception developed by J. Kevin O'Regan that gives a center role to actions in the development of visual awareness. The theory of perception proposed by Gibson [53] aims at answering the infamous question: "how do we see the world as we do?". The theory involves empirical research on real humans, a thorough description of the human environment, and how the individual experiences said environment.

SMCT and Gibson's visual perception theory differ on certain aspects of perception. In this manuscript, starting with inspirations and concepts from those two theories, we conceptualized and studied the sensory commutativity of action sequences. This paradigm aims at extracting information about the agent and its surroundings by comparing the different outcomes that result from playing an action sequence in different orders from the same starting point. Based on this, we developed application algorithms for object detection and self-discovery.

Developmental robotics

Although robotics is of course related to the problem of learning perception, it is not directly aimed at solving it. Robotics approaches therefore usually make use of a lot of knowledge about the robot and the environment, such as pre-programmed complex movements, positions (agents and objects), augmented visual sensors (depth cameras, 3D scans).

In the present thesis, we are interested in the emergence of perception from raw visual observations and primitive low-level actions, without the addition of engineered a priori knowledge. This is in line with a particular approach to robotics called developmental robotics which applies, among others, to the problem of perception. Developmental robotics [108], or epigenetic robotics, is a scientific field which aims at studying the developmental mechanisms, architectures and constraints that allow lifelong and open-ended learning of new skills and new knowledge in embodied machines. As in human children, learning is expected to be cumulative and of progressively increasing complexity, and to result from self-exploration of the world in combination with social interaction. The typical methodological approach consists in starting from theories of human and animal development elaborated in fields such as developmental psychology, neuroscience, developmental and evolutionary biology, and linguistics, then to formalize and implement them in robots.

1.4 Limitations as a starting point for undertaken research

While the current approaches have made significant progress in the last decade, the problem of perception is still largely unsolved for a number of reasons we list here. The hurdles identified here are starting points for the research and contributions presented in this thesis.

1.4.1 Problem simplification and the choice of simulations

In the problem of perception of embodied agents in human environments, the ideal platform would probably be a real robot, but for many practical reasons it is still very difficult to conduct long term experiments with real robots. A lot of study therefore resort to simulations, but the choice of simulation is crucial and this issue is currently largely overlooked.

For example, one of the most popular RL benchmark is the Arcade Learning Environment (ALE) [9], which consists of Atari 2600 arcade video games like Breakout or Pong. However, ALE is not intended to help create a multi-purpose learning robot by using Pong as a testbed. ALE is designed for developing "domain-independent" learning algorithms. The problem of perception is precisely domain-dependent: the world we live in is very specific and is governed by rules that living beings learn to understand in order to strive and our aim is to have artificial agents that understand our world, just like living beings do.

ALE is a single example of a larger trend. Most simulations that are used lack important features of the real life experience of a biological being. In ML: Mujoco [167], DeepmindLab [8], AI Habitat [147], etc... all miss crucial

features, and even in real robotics, robots that cannot displace themselves are extremely common. While these setups (robotics arms in general) reduce the problem to make progress, it is still hard to develop perception in these setups.

Research papers that propose methods and algorithms related to the problem of perception in which the simulations used do not satisfy basic criteria like first person partial observability, or coherent physics are useful for building the stepping stones with which more elaborated agents will be created. However, the missing features in the simulation used might also slow down progress on the core issues to face for real-life perception.

1.4.2 Task-specific knowledge

Compared to our generic objective of learning perception, RL algorithms only learn task-specific knowledge. When an agent is asked to solve a particular task in an environment, it specializes in this particular task, and is not able to learn general knowledge about the environment. This prevents the agent from re-using previous knowledge for solving new tasks rapidly, i.e. performing transfer learning. While many approaches in ML benefit from transfer learning, like neural network pre-training on ImageNet for Computer Vision applications, or Transformers pre-training for Natural Language Processing applications, this does not seem to be directly applicable to RL. We have yet to see any RL algorithm that shows such transfer learning properties.

The promise of State representation learning is closer to what we want as the learned features can depend on the environment, the perception and action capacities of the agent, but can be relatively task independent. However, most work forget the actions, and only learn using the observations. Actions and observations are tied, and understanding the contingencies is crucial for perception emergence, as many works have already suggested.

1.4.3 Technological locks

In addition to the aforementioned intrinsic problems of the current approaches, there are technological locks that prevent researchers from making progress.

A good example of this is Continual learning and memory with neural nets in general. The field still has a long way to go before providing useful tools. For now many propositions are still applied to prevent catastrophic forgetting on the MNIST dataset. The field is still lacking proper benchmarks to quantitatively assess performances, which slows down progress, even if recent propositions go in this direction [41].

Sample efficiency in RL is also an important issue. While RL does not learn general knowledge about the environment and how it works, it is still

the best approach the field came up with to this day for behavior learning. However the best applications still require a prohibitive amount of computation and time to learn such behaviour. Most algorithms require millions of timesteps in the environment before solving tasks.

1.4.4 Lack of development

Many of the above mentioned approaches are also lacking practical development.

For example, investigating the learning of disentangled representation looks promising, at least intuitively. However the proper definition of disentanglement is still debated. To this date, no definition has been widely accepted. Moreover, disentanglement is mostly viewed as another ML task, and thus suffers from the same design issue: it is based on offline static datasets, while the problem we tackle is online.

Approaches based on embodied agents theories of perception is a field with promising ideas that have been largely developed by psychologists and philosophers. It is currently under-developed from a modern computer science perspective. It needs theoretical and practical development in hard science to show its full potential.

1.5 Contributions

After setting up the problem of perception learning for artificial agents, we noticed the shortcomings of current approaches. In this thesis, we have made several efforts to overcome those shortcomings, and explore under-developed research directions. We, of course, did not solve the problem of perception (we would be much too rich to write this thesis), but we believe we asked relevant research questions and placed them in the context of the current research landscape. We hope that we shed light on novel scenarios and proposed interesting algorithms to make progress in promising directions.

The contributions of this thesis are:

- Novel approaches for state representation learning and continual learning for the embodied agent scenario.
- Theoretical contributions on a theory of disentanglement based on symmetries.
- Theoretical formalization of sensory commutativity for embodied agents: a novel approach for the problem of perception learning.
- Application of sensory commutativity: SCP, i.e. Sensory Commutativity Probability (automatic learning of the importance of each degree of freedom for an embodied agent).

- Application of sensory commutativity: SCOD, i.e. Sensory Commutativity Object Detection (active object detection for embodied agents).

1.6 Publications

Our work has resulted in the following publications:

1.6.1 Conferences

- **"On the Sensory Commutativity of Action Sequences for Embodied Agents"**, *H. Caselles-Dupré, M. Garcia-Ortiz, D. Filliat*, International Conference on Autonomous Agents and Multiagent Systems (AAMAS) 2021, Extended Abstract, Online. [27]
- **"Symmetry-Based Disentangled Representation Learning requires Interaction with Environments"**, *H. Caselles-Dupré, M. Garcia-Ortiz, D. Filliat*, Neural Information Processing Systems (NeurIPS) 2019, Vancouver, Canada. [26]
- **"Generative Models from the perspective of Continual Learning"**, *H. Caselles-Dupré*, T. Lesort*, M. Garcia-Ortiz, J-F. Goudou, D. Filliat*, International Joint Conference on Neural Networks (IJCNN) 2019, Budapest, Hungary. [95]
- **"S-TRIGGER: Continual State Representation Learning via Self-Triggered Generative Replay"**, *H. Caselles-Dupré, M. Garcia-Ortiz, D. Filliat*, International Joint Conference on Neural Networks (IJCNN) 2021, Online.

1.6.2 Workshops

- **"Object Detection for Embodied Agents using Sensory Commutativity of Action Sequences"**, *H. Caselles-Dupré, M. Garcia-Ortiz, D. Filliat*, NeurIPS 2020 Workshop on BabyMind. [27]
- **"On the Sensory Commutativity of Action Sequences for Embodied Agents"**, *H. Caselles-Dupré, M. Garcia-Ortiz, D. Filliat*, RSS'20 Workshop on Self-Supervised Robot Learning & ICML 2020 Workshop on Learning in Artificial Open Worlds. [28]
- **"Continual Reinforcement Learning deployed in Real-life using Policy Distillation and Sim2Real Transfer"**, *R. Traoré*, H. Caselles-Dupré*, T Lesort*, T. Sun, N. Díaz-Rodríguez, D. Filliat*, Workshop on "Multi-Task and Lifelong Reinforcement Learning", International Conference on Machine Learning (ICML) 2019, Long Beach, USA. [168]

- **"Symmetry-Based Disentangled Representation Learning requires Interaction with Environments"**, *H. Caselles-Dupré, M. Garcia-Ortiz, D. Filliat*, Workshop on "Structure & Priors in Reinforcement Learning", International Conference on Learning Representations (ICLR) 2019, New Orleans, USA. [26]
- **"Continual State Representation Learning for Reinforcement Learning using Generative Replay"**, *H. Caselles-Dupré, M. Garcia-Ortiz, D. Filliat*, Workshop on Continual Learning, Neural Information Processing Systems (NeurIPS) 2018, Montréal, Canada. [24]
- **"Generative Models from the perspective of Continual Learning"**, *H. Caselles-Dupré*, T. Lesort* M. Garcia-Ortiz, J-F. Goudou, D. Filliat*, Workshop on Continual Learning, Neural Information Processing Systems (NeurIPS) 2018, Montréal, Canada. [95]
- **"Flatland: a Lightweight First-Person 2-D Environment for Reinforcement Learning"**, *H. Caselles-Dupré, L. Annabi, O. Hagen, M. Garcia-Ortiz, D. Filliat*, Workshop on Continual Unsupervised Sensorimotor Learning, ICDL-EpiRob 2018, Tokyo, Japan. [23]

1.7 Outline

The rest of this manuscript is organized in the following chapters:

- Chapter 2 defines the problem of perception and introduces the hypothesis we use for our work on perception for embodied agents.
- Chapter 3 describes Representation Learning approaches for the problem of perception and our contribution on Representation Learning, Continual Learning and Reinforcement Learning for embodied agents.
- Chapter 4 presents our contributions to symmetry-based disentanglement in Representation Learning.
- Chapter 5 introduces our work inspired from perception theories on Sensory Commutativity and presents our theoretical contributions in Sensory Commutativity for embodied agents.
- Chapter 6 presents our practical approaches based on Sensory Commutativity: active object detection and improving sample-efficiency in RL.
- Chapter 7 takes a step back and discusses the merits and shortcomings of the research undertaken in the thesis, proposes future works and concludes.

Chapter 2

The problem of learning perception

Contents

In this chapter we define the problem of learning perception, and present the selected hypotheses used for our research in the context of this thesis.

2.1 Perception: problem definition and hypotheses

Perception is the medium by which agents organize and interpret sensory stimuli, in order to reason and act in an environment using their available actions [73]. Sensory stimuli usually come in various forms for biological agents: light, sound, physical contact, etc. In this thesis we focus on visual perception, i.e. the perception that emerges from visual sensory information and stimuli.

Thereafter, we propose the definition of the problem of visual perception considered in this thesis. All the contributions of the thesis aim at progressing in the understanding and learning of perception in this sense.

2.1.1 Perception problem definition

Perception is a puzzling concept that has been widely studied by philosophers, psychologists and cognitive scientists. How come we experience life and reality as we do? How do babies go from relatively limited and helpless beings to intelligent adults? These questions are intriguing to humans in general, and to researchers in AI in particular, as answering these questions could hold part of the answers of the question of the emergence of artificial intelligence.

Philosophers developed thought experiments to characterize perception and its role in our experience of life. Psychologists have developed theories supported by empirical evidence from experiments with humans or animals to propose models of perception. Today cognitive sciences and neuroscience are developing this path by experimenting on a more deeper level the mechanisms of perception in our brains.

These research fields aim at explaining how high-level concepts are formed by the brain using vision. From the high-dimensional sensor apparatus we have in our ocular system, we are able to extract meaningful and actionable information about our surroundings. We also observe that animals and humans are able to transfer to new settings, and we assume that it is because they acquire transferable knowledge and skills (such as, for example, common sense, a notion of intuitive physics, of objects, ...). This knowledge supposedly allows humans and animals to make sense of the world, and this common-sense allows for generalization across a huge variety of situations. Humans build on this knowledge to create even more advanced knowledge by combining other skills like social communication, counter-factual thinking, etc.

This knowledge is obvious to us and biological beings in general, because we *are* well-designed learning agents. The research question is thus: how can an artificial agent learn common sense?

Hence, there are a number of hypotheses we have to state in order to study the problem of perception. We will use them after for our contributions on the study of perception.

2.1.2 Hypothesis: unsupervised learning

In order to build a multi-purpose robot, a naive idea would be to hard-code it to solve all the tasks we want it to solve. However, it is generally not feasible to do so: a program to solve all the tasks in the real-world would be incredibly complex. Sensors are high-dimensional, classical locomotion and manipulation approaches require a model of the body for planning, and the world itself has too many interactions to take into account (objects of all sizes and shapes, a large range of terrains and landscapes, etc). The learning paradigm is a natural way of alleviating those difficulties to create agents

that can reason and act in the world. In addition, biological agents do learn, and it is the only successful example of intelligence we can observe.

Supervised learning, where humans use labeled data to create software that can solve tasks is an approach that has many applications such as face recognition, automatic diagnostics or sorting in supply chains. However it is not feasible to use this technique for learning generic perception because of the compositional nature of the world and the variety of the environment: you can always encounter a scene that you never encountered before. The technique is not applicable to learning common sense and to the emergence of agents with strong generalizable skills. Common-sense is associated with meaning, and there is no meaning without grounding the knowledge in real experiences.

In order to be able to adapt quickly to any task, agents have to learn on their own or from a weakly supervised source of data. For instance children learn by themselves with few, but extremely helpful and precise, feedback from their parents. We will consider agents that are situated in environments in which they can act through motor commands, and simultaneously receive observations using their sensory apparatus. Using this stream of data and possibly extraneous information or feedback, we want agents to autonomously learn relevant and re-usable information for task solving like navigation, manipulation, identification of objects, etc.

2.1.3 Hypothesis: agents and environments

For the problem we consider, there are requirements that need to be met in experiments so that we are able to study visual perception. In particular, we only consider a specific set of agent-environment setups for our study of perception, which we will justify thereafter.

First, an agent should be able to actively select the data it sees [53], a procedure often called active perception. This is linked to research [53, 131, 134] that argue that perception and common sense emerges from the relation between sensors and actuators. Perception is not something that has an objective reality outside of the agent, but only a byproduct of an agent trying to make sense and survive using its sensors and actuators.

The second aspect is the environment of the agent. In all of the theories of perception mentioned before, it is a critical aspect often studied deeply. The theories aim at explaining the emergence of perception from the relationship between the agent and its environment. The very nature of the environment; its rules, properties and characteristics are precisely what the philosophers/psychologists/cognitive scientists exploit to explain perception. A change in the environment rules and properties could radically change perception, and thus life as beings know it. The intelligence and skills learned by agents are conditioned and constrained by what the agents can perceive. Poincaré [134], for example, already mentions it in a manuscript

in 1895 where he gives an example of a world where light properties are different, with the supposed consequences that humans would perceive the world as 4D.

We thus consider an agent-environment experimental setup as viable if different required features are present. The agent should have a body with enough degrees of freedom to allow large-scale movement like navigation and also finer skills like manipulation. The agent should have partial observations of the environments, using embodied sensors (like eyes/camera). The environment rules should be similar to those of our world: the world is composed of objects that obey physical laws. Objects can be movable or not, collide with each other and agents. The environment should respect the rules of temporal continuity and causality, ... We call this the embodied scenario, illustrated in Fig.2.1. The goal is to understand, predict, analyze the sensorimotor flow of data that an embodied agent receives.

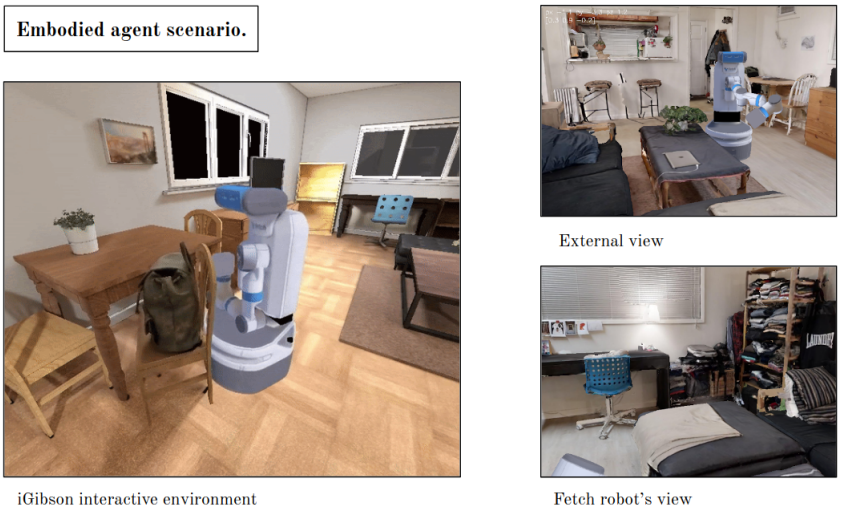


Figure 2.1: Example of an embodied agent. The robot sees the world through its embodied camera, and has 10 degrees of freedom that allow navigation and manipulation.

2.1.4 Hypothesis: innate vs acquired

A robotic agent facing deployment in the world would need to learn from their own experience and continually update their knowledge about the world. The world changes over time, and our ability to accommodate those changes is a crucial aspect of our survival and prosperity. Then, equipping the agent with pre-computed knowledge of the world doesn't seem suited for our ever-changing and unpredictable world. By adding prior and assumptions in the system, you project your own understanding and pre-conceived

ideas about how perception occurs. This is a double-edge sword, as you greatly reduce the search space, but you might reduce it too much and constrain your system too much. This is why we tend to limit the a priori knowledge given to the agent. This concerns the sensors and motors of the agent.

About sensors, we aim at making perception emerge from the same sensors that receive biological agents. Robots in uncharted environments can only rely on particular sensors (rgb camera, lidar, depth, ...). The most stable is RGB images as observations, and it is also the closest to the human sensor. We choose to focus on this one for our experiments, however the underlying principles hold independently on the kind of sensor you would consider. Hence, we do not give additional information in the observation received by the agent, such as direct access to depth, position in the environment, or any information that a biological agent might not have. We acknowledge that this hypothesis is not necessary: we could construct better agents using additional information that, whether we like it or not, is at our disposal. However, this hypothesis is interesting to apply in our case since we want to understand perception from purely visual information. In this case it is relevant to restrain the observations of the agent to RGB-images in order to emulate a scenario encountered by a real robot in the world. The sense of depth for instance could be learned from interaction with the environment. In some case, we decide to equip the agent with a basic ability, like the ability to compare images and telling if they are the same or different, see our work on object detection using sensory commutativity of action sequences (Chapter6). We consider this rather primitive ability as something not necessary to learn, but just a mechanism that allows learning.

About motors, we believe that the agent should learn to control its body on its own and not receive precise geometric models or pre-defined commands for high level actions like grabbing items for manipulations. Babies and animals have to learn to control their own bodies, and this learning experience definitely helps them understand the world [36, 76] and how it functions. We thus restrict the movements that the agent can already do when spawning in the environment.

2.1.5 Related work

We illustrate our hypotheses and our research by presenting some related work that achieves progress in the quest of agent perception.

The learning of affordances is a good example of perception learning in artificial agents. Affordances is a term created by Gibson [53] that describes the opportunity for action that elements in the environment provide to an agent. Depending on the situation, an object can have different affordances, for instance a chair can be used as a seat or as something to stand on to reach for objects that are high in altitude. Affordances require a relationship

in which the environment and the animal can work together. Recent work have focused on creating labeled dataset for affordance learning [117, 115], and with that came several deep learning algorithms that provide methods for the automatic inference of affordances [116, 87, 39, 30, 46, 181]. By leveraging the capability of novel deep learning algorithms to map images to classes, researchers are able to estimate affordances based on images of an object, eventually leading to the implementation of new perceptual skills for artificial agents.

Another example is creating an agent's understanding of the 3D structure of the environment. By using work by Poincaré and O'Reagan [134, 123] that suggest that the Euclidean structure of space induces invariants in an agent's raw sensorimotor experience, Laflaquière et al. [90] use the ML paradigm and neural networks to learn a 3D understanding of the world using raw observation of an agent, in an unsupervised fashion. While these insights have been present in the literature for decades, even centuries, the advent of ML research has allowed novel useful implementations of these ideas.

In the same vein, we have research work that takes inspiration from the study of intrinsic curiosity among children for developing learning mechanisms for autonomous behaviour and task solving [124, 74, 7]. Babies are known to play with their environments not randomly, but by trying to autonomously create order, like stacking cubes in a playground [6]. They create goals for themselves autonomously, and cognitive sciences have discovered that this learning mechanism helps them develop their perception of the world. Several implementations of this idea have been made in artificial systems [128, 32].

2.2 Experimental setups

We now describe the different experimental setups that are available for implementing algorithms for the problem of perception. In order to test and validate our idea, we need to pick from the vast possibilities offered in the research landscape in terms of robots, simulators, tasks, environments and agents. We first review the (real) robots setup and explain why it is a problem currently for ML research. We then describe the simulators available at the moment, and we use our hypothesis presented above to select which one seems appropriate for the scientific questions of perception we would like to answer.

2.2.1 Real robots, a problem

The best experimental setup for the problem of perception is obviously a real-life setup. Having a real robot setup, where the robot can move and

manipulate the environment just like a biological agent, is costly and complex. First of all, the state of the art in robot manufacturing shows that having a high-quality robot, with many Degrees of Freedom (DOF), precise movements and development features is a monumental challenge in itself. Low-cost robots are often buggy, fragile and thus cannot really be used for ML research. High-quality robots are usually expensive (X00,000$+) and still very slow in general, in order to avoid damages. There exist cheap low-end robots that allow for cheaper experiments but are limited in what they can accomplish, like the Pepper robot of SBRE. Recently, Boston Dynamics have shown impressive progress with the Spot robot, which is more affordable given how high-end it is. They also showcased communications videos of humanoid robots that look impressive but it is still unsure if these products will be commercialized and engineered for facilitating AI research.

There is a need for cheap robots with enough DOF for complex interactions with their environment, such as unconstrained manipulations in human environments. This is a challenge faced by the robotics industry, and even if companies like Softbank Robotics Europe have tried pushing in this direction with Pepper and Nao robots, there is still a long way to go before having a proper AI robotic benchmark. In our thesis we were planning on experimenting with the Pepper robot but we were not able to do so because of the limitation of the robot software and what it can accomplish in terms of navigation. This issue inhibits progress towards real-life applications of robots. With a common benchmark of proper perception tasks on the same accessible robot, the ML research community could make tremendous progress, just like we saw with the ImageNet challenge (resulted in CNNs and all computer vision applications), NLP benchmarks (resulted in Transformers and all NLP+computer vision applications), protein folding competitions (resulted in AlphaFold which has many potential applications).

In practice, in classical robotics, we often use a controlled robotic setup with a robotic arm or a mobile robot where we utilize external information about the agent and the environment, such as position, joint parameters, object positions, and annotated data. This allows the experimenter to distill its knowledge in the form of priors into the system (e.g. knowledge of the workspace in the case of a robot interacting with objects on a table). However, this information might not be available in the general case. Instead, in nature, children and animals do not have access to this information when they are born. They start from a relatively naive setup, and then build perception via interaction with the environment. We aim at developing theories and applications for this tabula-rasa case where the agent is naive: it can only actuate its motors (without any description of what they do) and receive observations through its sensors.

All of these issues make the real-life setup hard to work with. That is why most researchers in agent perception choose to use simulations to

accelerate progress before moving on to real-life deployment. This is also linked to the fact that we are dealing with statistical learning, which requires huge quantities of data, thereby increasing again the stress on the platforms.

2.2.2 Simulations

We now describe the simulation benchmarks used in ML research for the problem of perception, and divide them into two simple categories: the ones that do satisfy the requirements outlined above for studying the problem, and the ones that do not.

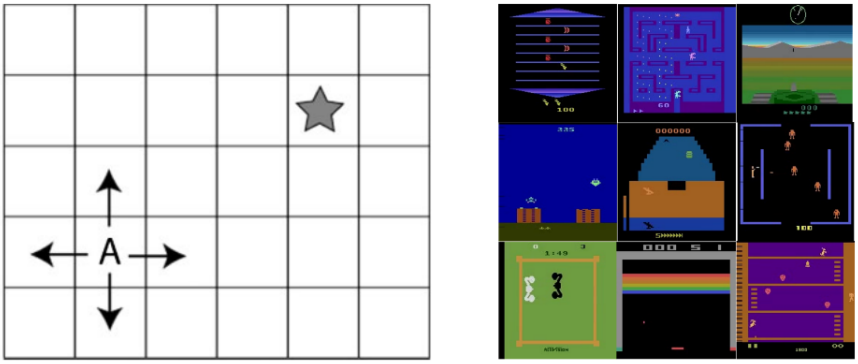


Figure 2.2: **Left**: a grid world environment. **Right**: Atari 2600 arcade video games.

Grid worlds

The simplest environments, which are often used for prototyping algorithms and proof of concepts, are called grid worlds. Those worlds are populated by virtual agents that do not even have a body (only a mathematical entity equipped with high-level actions), but are rather incarnated by a position on a given finite 2D map of states. The agent can move discretly on the map. Depending on the implementation, events can occur when the "agent" visits a state, like rewards or penalties.

These environments are useful for understanding novel concepts and visualization, but they cannot possibly be acceptable benchmarks for the problem of perception, for obvious reasons (discrete, no agent). We illustrate an example of Grid World in Fig.2.2 (left). The agent is defined by its position on the map and can *move* up, down, left or right. It has the goal to reach the star position.

Retro video games

A very popular RL benchmark is the Arcade Learning Environment (ALE) [9], which consists of Atari 2600 arcade video games like Breakout or Pong. Retro video game environments are rampant: VizDoom [82] (Doom game), Retro Gym [119] (30 SEGA Genesis games) or OpenAI Gym [16] (retro-like games). We illustrate the Atari suite of games on Fig.2.2 (right).

First, all of these environments have agents ruled by high-level actions such as jump or go left/right. Reality is not quite as easy: biological agents have to learn to master their motor commands in order to do crucial fine manipulation of their environment that allows, for example, to detect moving objects. This is not modeled after retro games. Moreover, most of these worlds are 2D and do not possess the same basic properties of real-life such as first person view or movable objects.

Control environments

Usually developed for robotics setup, control environments aim at learning fine manipulation with a complex body such as a legged robot or a robotic hand. Simulators include Mujoco [167], DeepMind's Control Suite [162], PyBullet [37]. Fig.2.3 illustrates the MuJoCo suite (left), with 4 different locomotion tasks. Each agent has a different body and needs to learn to displace it.

All of them allow precise simulation of realistic movements. However environments are nonexistent. The agents are placed in void spaces, with one or two possible interactions with exterior elements at best. Moreover, the problem is often to learn a policy that maximizes an outcome like reaching a position fast. The agent is given its position and motor commands. There are no sensory observations, no image from the eye. This is why the agent cannot learn anything about interaction with the world, and thus cannot build perception.

3D mazes with low action space

These environments refer to a 3D world where the agent experiences the world in first-person, but does not have a proper body with several motors. Its actions are restricted to move left/right, jump, etc. These environments are often used for studying navigation and reasoning. Examples are DeepMind Lab [8], VizDoom [82] or AI Habitat [147]. Fig.2.3 illustrates the DeepMind Lab environment (left), where the agent needs to collect apples in a maze.

For the same reasons as retro games, since the agents do not have any bodies, these simulators are not suitable for studying the problem of perception. Agents should learn to coordinate low-level actions to create high-level actions. Otherwise fine manipulation of the world is not possible.

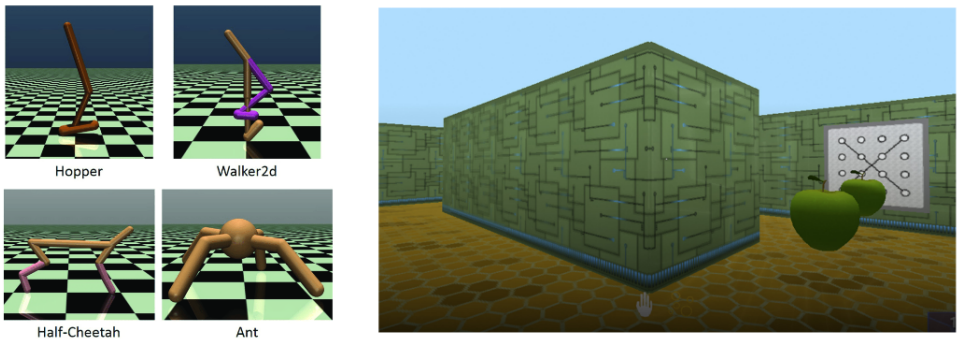


Figure 2.3: **Left**: MuJoCo control tasks. **Right**: the DeepMind Lab environment.

3D realistic environments

iGibson is a simulation environment for robotics providing fast visual rendering and physics simulation. It is packed with a dataset with hundreds of large 3D environments reconstructed from real homes and offices, and interactive objects that can be pushed and actuated. The simulator offers a wide variety of realistic agents with numerous DOF, like the Fetch robot. Fetch is originally a 10-DOF real robot [174] equipped with a 7-DOF articulated arm, a base with two wheels, and a liftable torso. In iGibson and in real-life, Fetch perceives the environment through a camera placed in its head. The environment and agent setup is illustrated on Fig.2.1.

iGibson is very well suited for studying the problem of perception, because of how it models most problems a biological agent faces when learning to manipulate its world. Still, it has many bugs, which is expected as it is currently being developed and improved. Also, the speed of the simulation can be prohibitive for large-scale experiments, such as million-frames RL experiments.

2D *realistic* environments

To alleviate the problem of computing power requirements, we developed Flatland [23]. It is a simple, lightweight environment for fast prototyping and testing. It is of lower complexity compared to similar 3D platforms (e.g. iGibson), but emulates physical properties of the real world, such as continuity, multi-modal partially-observable states with first-person view and coherent physics. We propose to use it as an intermediary benchmark for problems related to perception. Flatland is highly customizable and offers a wide range of task difficulty to extensively evaluate the properties of artificial agents. An example of a Flatland environment is illustrated on Fig.2.4. The agent has two 2-DOF arms, a base that can spin and move

forward or backward, and a head fixed on the base that can pivot. The agent observes a field of view that is in front of it. It can interact with moving, movable, or fixed objects, and activate traps or rewards.

Usually, in our experiments, we first prototype our algorithms on Flatland, and then move on to iGibson to show that they are robust to more complex realistic environments.

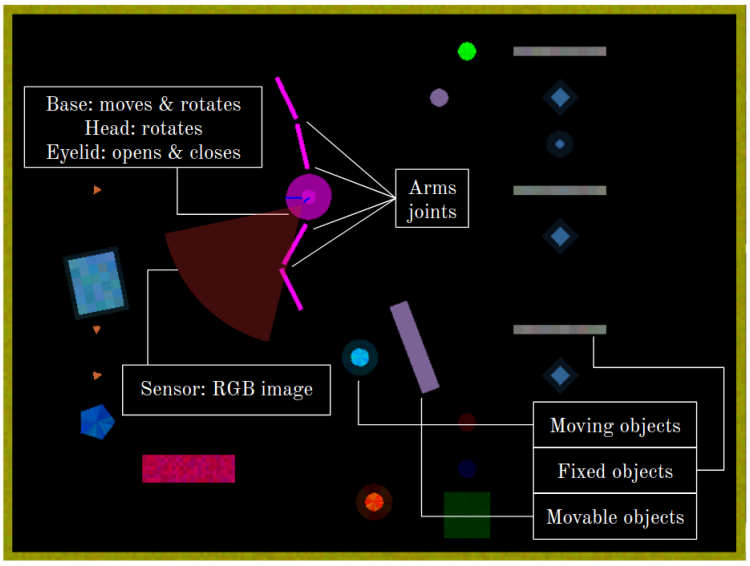


Figure 2.4: The Flatland environment is an example of a 2D experimental setup suited for studying the problem of perception.

2.3 Conclusion.

In this chapter we described the setup and context for the problem of learning perception in artificial agents and the main hypothesis we believe are important for studying this problem. We then reviewed the potential platforms for such work and we focused on 2D and 3D environments that have the requirements for studying the problem. We believe it is an important step towards creating agents that learn to face the difficulties of a real-life setup.

Now, Chapters 3 to 6 will present the different approaches and algorithms that have been developed and investigated in this thesis.

Chapter 3

Representation learning for perception and applications

Contents

In this Chapter, we introduce the field of Representation Learning. Representation learning is a broad term for all learning methods that create intermediary representations of data, which can be used to solve tasks efficiently. This approach has received an exponentially rising amount of interest in the last decade, as Deep Learning basically provides efficient methods for Representation Learning from raw data. Representation Learning can thus also be applied to the problem of perception, since biological agents learn abstract concepts from raw data and learn to associate them. When applied to an agent-environment setup, Representation Learning methods are often referred to as State Representation Learning methods (SRL).

We now provide an overview of the successful approaches in Representation Learning for agent perception, and then present our contributions to this field.

3.1 Background on Representation Learning methods

3.1.1 Image models

The promise of deep learning is to discover rich, hierarchical models that represent probability distributions over the kinds of data encountered in artificial intelligence applications, such as natural images, audio waveforms containing speech, and symbols in natural language corpora. In this thesis, we focus on vision approaches. These models are divided into discriminative models and generative models.

Discriminative models

Discriminative models aim at classifying data into various categories. This has historically been done with models like bag of features associated with SVMs [18]. Then came the Deep Learning revolution, where the advent of neural networks help automatically determine important features of data for classification.

Neural networks learn by processing data examples, each of which contains a known "input" and "output," forming probability-weighted associations between the two. The training of a neural network from a given data example is usually conducted by determining the distance between the processed output of the network (the prediction) and the target output, the value of this distance being the error. As deep Neural Networks models are fully differentiable, we can calculate the contribution of each parameter to the error, and change the value of the parameters in the direction which

diminishes the error (gradient descent). Successive adjustments will cause the neural network to produce output which is increasingly similar to the target output. After a sufficient number of these adjustments the training can be terminated based upon certain criteria. This is known as supervised learning. Among many hyper-parameters, the architecture of the neural network is crucial for the quality of training. The basic architecture is a neural network where all neurons from one layer are connected to all neurons of the previous and next layer: this is called a Multi Layered Perceptron (MLP), illustrated on Fig.3.1.

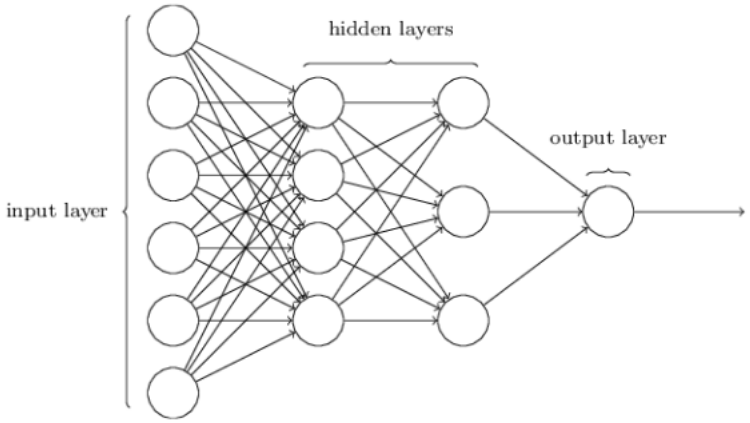


Figure 3.1: A Multi Layered Perceptron with 2 hidden layers.

Convolutional Neural Networks (CNNs) depart from this basic architecture to leverage local properties of images and hierarchical combinations of features. Instead of connecting all neurons from a layer to all previous and next layers, they use convolutions to inspect patches of images locally, and then slide this convolution across the input images. Multiple layers of such convolution allow the construction of hierarchical representations of images more easily since images are spatially coherent. The architecture is illustrated on Fig.3.2.

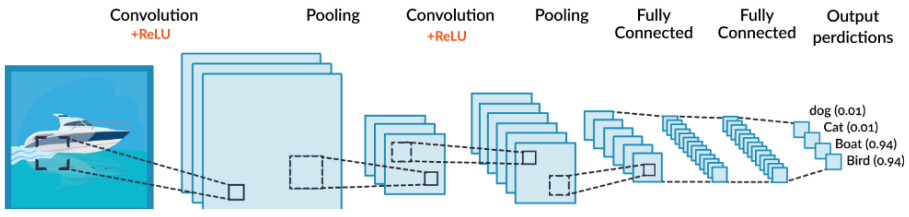


Figure 3.2: A Convolutional Neural Network.

Around 2012, CNNs enjoyed a huge surge in popularity after a CNN achieved state-of-the-art performance labeling pictures in the ImageNet chal-

lenge. Since Alex Krizhevsky et al. published the paper "ImageNet Classification with Deep Convolutional Neural Networks" [89] describing the winning AlexNet model, a plethora of research has extended the breakthrough, with notable models like ResNet [66] for image classification or YOLO [13] for object detection. Throughout the past several years, CNNs have achieved excellent performance describing natural images (including ImageNet [38], COCO [100], and VisualGenome [88]), performing facial recognition (including CelebA [102]), and analyzing medical images (including chest x-rays and photos of skin lesions).

Discriminative models like CNNs learn features about the images that are related to interesting semantic information about the data, but they require enormous amounts of labeled data for training. However, these features can be re-purposed for other related tasks where less data is available, which has extensively been done with pre-trained image models on ImageNet, and with great success.

Generative models

Contrary to discriminative models, generative models aim at modelling the distribution of the data source in order to be able to generate new data examples. Learning a generative model is a complex task that supposedly requires understanding abstract concepts in the data, quite like the famous quote says: *what I cannot create I cannot understand*. From pretrained generative models, we can supposedly extract those abstract concepts about data, which could be used by an agent to understand its surroundings and improve its perception of the world. We now present the two most popular frameworks for generative models, Variational Auto-Encoders (VAEs) and Generative Adversarial Networks (GANs).

VAEs [84] are a particular kind of auto-encoder that learns to map their input into continuous latent representation, in order to learn the marginal likelihood of the data x given the latent representation z. This is illustrated on Fig. 3.3. VAEs learn to reconstruct the input data by compressing it into the latent space that is of much lower dimension. The optimization process thus aims at copying the input data by only having access to a *summary* of it.

This optimization can be written as the sum of two terms, which then can be used to derive a tractable lower bound objective to the true objective called the Evidence Lower Bound (ELBO):

$$\log(p_\theta(x|z)) \geq ELBO(\phi, \theta, x, z) = \mathbb{E}_{q_\phi(z|x)}[\log(p_\theta(x|z))] - D_{KL}(q_\phi(z|x)\|p(z)) \tag{3.1}$$

where D_{KL} is the Kullback-Lieber Divergence between the true and approximate posterior, and ϕ, θ respectively represent the weights of the encoder and decoder (see Fig. 3.3). In practice, this objective is maximized

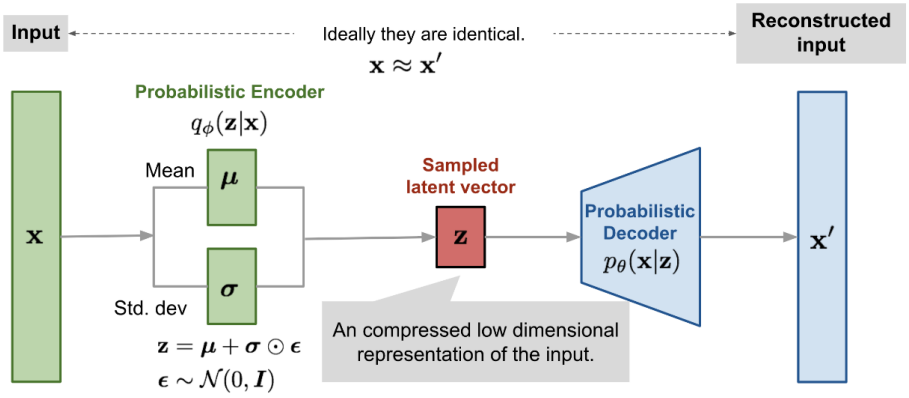


Figure 3.3: The Variational Auto-Encoder architecture.

using assumptions: the prior $p(z)$ and posterior $q_\phi(z|x)$ distributions are parametrized as isotropic unit Gaussians, and the so-called "reparameterization trick" is used to estimate the gradients of the objective function. VAEs also have been extended to Conditional-VAEs (or CVAEs) to support class-conditional generation. More elaborated versions of the VAE have been proposed such as VAEs that learn disentangled representation of data like Beta-VAE [69] or CCI-VAE [19], which we use in our research. In those variations, authors change the loss function of the VAE to induce the learning of a constrained latent space, which they show can help to learn representations that are more interpretable and thus better suited for downstream tasks.

GANs [56] are a framework of generative models, where the learning process is a minimax game between two neural networks, a generator G and a discriminator D. The goal of D is to discriminate between the real and generated (i.e., fake) data, whilst the goal of G is to map a noise distribution to the real data distribution. The architecture and training process is illustrated on Fig.3.4.

Generated samples aim to maximally confuse the discriminator, through the following objective function:

$$\min_G \max_D \mathbb{E}_{x \sim p_{data}}[\log D(x))] + \mathbb{E}_{z \sim p_z}[\log(1 - D(G(z)))], \tag{3.2}$$

where p_{data} and p_z are the real data distribution and noise distribution, respectively.

First introduced in [56], many follow-up works have extended and improved upon the original model. Among these, GANs have been extended to Conditional-GANs (or CGANs) to support class-conditional generation [112], and a plethora of papers [3, 12, 120, 58] have focused on modifying the objective function (eq. 3.2) to stabilize training and improve the generation quality. One of the model we evaluate, the Wasserstein GAN (WGAN)

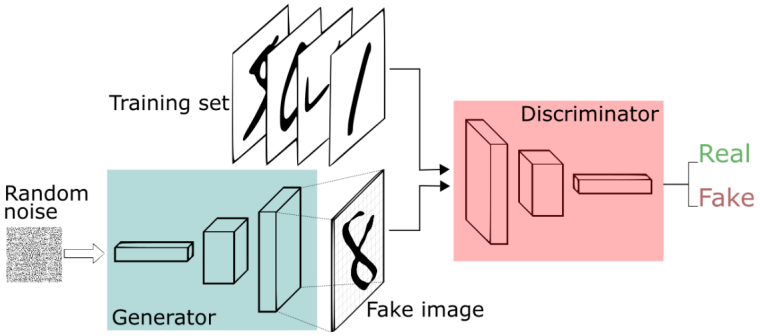


Figure 3.4: The Generative Adversarial Network architecture.

[3], try to address training issues by enforcing a Lipschitz constraint on the discriminator, which stabilizes training and avoids mode collapse (a collapse problem where the GAN only generates slight variation of one single image).

Generative models are commonly used for State Representation Learning, as they can successfully learn compressed and useful representations of visual images. When trained on an agent's sensory inputs, they provide a vectorial representation of what the agent experience. They have recently shown promising results as State Representation models for RL [60, 171]. They can also generate data, i.e. generate sensory states that the agent could have experienced.

3.1.2 Forward and inverse models

Another class of algorithms that are more directly applicable to the environment-agent scenario are forward and inverse models. They are particularly interesting because they can be trained using self-supervision using only sequences of images and actions recorded as the agent acts, without any human annotation.

Forward models aim at predicting future states given current and past states, as well as current and past actions. Thus, the model learns the dynamics of the world, as the agent experiences it. As a byproduct of this learning, we can hope that the representations learned for prediction are relevant for perception.

Simple feed-forward (in the fully observable case) or recurrent networks (in partially observable cases where memory is required) [96] can in principle be implemented to learn the dynamics of the world. However, in most cases they are not expressive enough to model the complex interactions that exist in the world's dynamics, especially when the scenario is realistic. Hence, researchers have been developing novel approaches for forward models, with the intent of building a general-purpose forward model that can be applied

to various scenarios.

For modelling the dynamics of the world, there usually needs to be a memory component that can remember the particular setup and events that took place in the environment. If the agent-environment setup is Markovian like in the MDP framework, memory is not needed.

World models [60] is an approach that combines generative models and memory architecture to create vision and memory modules that form a forward model when combined. Other approaches have been developed for particular benchmarks such as Atari [9], Chess, Go and Shogi [150]. Most recently, a single architecture of the forward model has been successfully trained on all those benchmarks [77]. Using the learned model, planning can be applied to solve the benchmarks, showing the powerfulness of such an approach. The real problem is them shifted to building a forward model, not solving the task.

For embodied agents however, less progress has been made. There is no general well performing architecture that works on a setup with an agent that has multiple degrees of freedom and is placed in a realistic environment. The closest we have is MERLIN [171], which is an architecture that learns a forward model for a body-less agent in realistic mazes and rooms. The result is that the agent is able to memorize and reason in its world, and thus solve complex tasks.

Inverse models are designed to take two consecutive states in an agent-environment setup and predict the associated action that leads to the transition between the two states. These models help understand the consequences of actions in the environment, and thus help understanding how the agent interacts with the environment. They are usually more simple than forward models, as the output (actions) is of lower dimension than for forward models (states or images), but also make it possible to learn relevant features.

3.1.3 Reinforcement learning algorithms

The most used environment-agent setup is the RL framework. In this framework, the agent has a set of actions $\mathcal{A}$ and the set of possible states of the world is noted as $\mathcal{S}$. Regarding the dynamics of the world, given a state $s \in \mathcal{S}$ and an action $a \in \mathcal{A}$, the probability of transitioning to another state s' is noted $\mathcal{P}_a(s', s) = \mathbb{P}[s'|s, a]$ and forms a transition matrix $\mathcal{P}_a$. On top of this, each transition is associated with a (possibly null) reward noted $\mathcal{R}_a(s', s)$ and forms a reward matrix $\mathcal{R}_a$. The whole framework is called a Markov Decision Process (MPD) $(\mathcal{S}, \mathcal{A}, \mathcal{P}_a, \mathcal{R}_a$ because each transition and reward only depends on the last state and not the whole trajectory, i.e. it has the Markov property. The associated problem with this framework is expected reward maximization. It asks: what is the policy, i.e. a function $\Pi(s \in \mathcal{S}) \in \mathcal{A}$, that gives for each state the action maximizing the expected

discounted accumulated reward:

$$\max_{\Pi} \mathbb{E}[\sum_{t=0}^{\infty} \gamma^t \mathcal{R}_{a_t}(s_t, s_{t+1})]$$

This framework is general because it can model environment-agent worlds of various nature: games, grid-world, control problems. Hence, efficient algorithms for this problem can in theory be directly applied to all scenarios modeled by the MDP.

The development of algorithms for solving the RL problem has bloomed in recent years, thanks to the development of Deep Learning. Powerful function approximators directly find application in RL, because of mathematical objects that have been constructed to answer the RL problem. The value function and Q-function are essential tools for finding optimal policies. Intuitively, the value function estimates "how good" it is to be in a given state, while the Q-function estimates the quality of a state–action combination with respect to the reward. From the approximation of those functions we can derive policies that provably converge to optimal policies that maximizes the reward signal. DL has brought powerful function approximators that can estimate these functions and thus estimate well-performing policies. Another type of approaches are policy-based methods that try to directly estimate the optimal policy.

Whether they use a value-based [113, 67] or policy-based [151, 61] algorithm, the function approximators are always deep neural networks that learn representation of the input data hierarchically, so RL algorithms almost always qualify as Representation Learning algorithms.

Regarding the inputs given to RL algorithms, there are two approaches: learning from raw observations or learning from learned representations: an approach called State Representation Learning (SRL) [96]. In SRL, a model that takes raw observations as input and output representation is trained without supervision on observations collected by the agent. Then, this model is used as input to the RL algorithm to learn a policy. The goal of SRL is to learn a powerful representation model of the world, for example using a forward model, before trying to solve particular tasks with RL benefiting from the transfer of the learned representation. The SRL should therefore learn general knowledge about the world and its dynamics so that it can be re-used for a wide range of tasks.

3.1.4 Continual Learning

Learning in a continual fashion is a key aspect for cognitive development among biological species [45]. Humans have a remarkable ability to continually learn and adapt to new scenarios over the duration of their lifetime [158]. This ability is referred to as never ending learning, also known as continual learning or lifelong learning. Never-ending learning is the constant

development of increasingly complex behaviors and the process of building complicated skills on top of those already developed [143], while being able to reapply, adapt and generalise its abilities to new situations. The promise of CL is thus to create agents that learn behaviors and skills while solving its tasks and reuse these skills later as stepping stones.

In Machine Learning, such a learning scenario has been formalized as a Continual Learning (CL) setting [160, 118, 154, 156, 153]. The goal of CL is to learn from a data distribution that changes over time without forgetting crucial information. Unfortunately, neural networks trained with back-propagation are unable to retain previously learned information when the data distribution changes, an infamous problem called "catastrophic forgetting" [49]. Successful attempts at CL with neural networks have to overcome the inexorable forgetting happening when tasks change.

There are 4 main approaches for continual learning:

The first approach, referred to as *rehearsal*, is to keep samples from previous tasks. The samples may then be used in different ways to overcome forgetting. The method can not be used in a scenario where data from previous tasks is not available, but it remains a competitive baseline [140, 118]. Furthermore, the scalability of this method can also be questioned because the memory needed to store samples grows linearly with the number of tasks.

The second approach employs *regularization*. Regularization constraints weight updates in order to maintain knowledge from previous tasks and thus avoid forgetting. Elastic Weight Consolidation (EWC) [85] has become the standard method for this type of regularization. It estimates the weights' importance and adapt the regularization accordingly. Extensions of EWC have been proposed, such as online EWC [153]. Another well known regularization method is *distillation*, which transfers previously learned knowledge to a new model. Initially proposed by [72], it has gained popularity in CL [99, 140, 176, 156] as it enables the model to learn about previous tasks and the current task at the same time.

The third approach is the use of a *dynamic architecture* to maintain past knowledge and learn new information. Remarkable approaches that implement this method are Progressive Networks [144], Learning Without Forgetting (LWF) [98] and PathNet [47].

The fourth and more recent approach is *generative replay* [156], where a generative model is trained alongside each task, and used when a new task arrives to produce samples from previous tasks in order to not forget them. This approach has also been referred to as *pseudo-rehearsal*.

3.1.5 Contributions to the field

In the remainder of this chapter, we will present two contributions we made to the Representation Learning field during the thesis. The first one is S-

TRIGGER, which aims at proposing a model for building a state representation model for control, in a continual learning setting using the *generative replay* approach. The second one is DisCoRL, a method for RL in a continual learning setup, with applications to real robots using the *distillation* approach.

3.2 Contribution: S-TRIGGER

3.2.1 Abstract

We consider the problem of building a state representation model for control, in a continual learning setting. As the environment changes, the aim is to efficiently compress the sensory state information without losing past knowledge, and then use Reinforcement Learning on the resulting features for efficient policy learning. To this end, we propose S-TRIGGER, a general method for Continual State Representation Learning applicable to Variational Auto-Encoders and its many variants. The method is based on Generative Replay, i.e. the use of generated samples to maintain past knowledge. It comes with a statistically sound method for environment change detection, which self-triggers the Generative Replay. Our experiments on VAEs show that S-TRIGGER learns state representations that allow fast and high-performing Reinforcement Learning, while avoiding catastrophic forgetting. The resulting system has a bounded size and is capable of autonomously learning new information without using past data.

3.2.2 Introduction and contributions

The long term goal of creating agents capable of learning during extended periods of time, without constant supervision, will require the capability of continually learning about their environment. Indeed, learning about the world is inherently a progressive/incremental task, since the data distribution experienced by the agent can change both spatially and temporally. This involves building a model of an agent's surroundings, with visual features, and continually updating this model as its life progresses and the environment evolves, without forgetting.

Recent advances in representation learning [10] have brought successful tools for learning a model of the surroundings of the agent, a field known as State Representation Learning (SRL) [96]. Once trained, these representation models are used to learn policies using Reinforcement Learning (RL) more efficiently [60] than when using raw sensors. However, previous approaches have mostly been limited to scenarios where the environment stays fixed. Since the models used are often neural networks trained using stochastic gradient descent (or any variant), they forget past knowledge when the training data distribution changes. In Continual SRL, the training

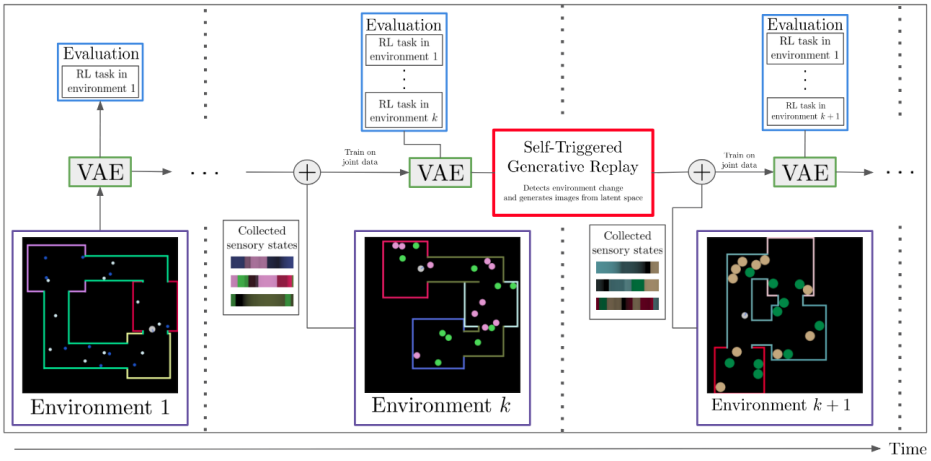


Figure 3.5: Overview of our proposed method S-TRIGGER for Continual State Representation Learning for RL. As the agent lives in environment 1, it collects sensory states (i.e. vision) to learn a state representation model (here, a VAE). We test this model by using it to solve, in an on-policy fashion, a RL task in environment 1. The agent is now moved to environment 2. Our method automatically detects this change, and uses the VAE trained on environment 1 to generate images corresponding to environment 1, that are joined to collected sensory states of environment 2 to learn a state representation model for both environments. We test this state representation model by learning, in a on-policy fashion, to efficiently solve a task in environment 1 **and** 2. The procedure continues as the agent encounters new environments. Best viewed in color.

data distribution changes as the environment evolves, which can happen in a discrete or continuous way. In our experiments, we consider sudden discrete changes of environment.

We propose a general method for Continual SRL for agents behaving in a changing environment, termed Self-TRIGgered GEnerative Replay (S-TRIGGER for short), with the aim of learning representations useful for RL. Our approach is compatible with any type of State Representation model that is generative and reconstructs the input, which includes Varational Autoencoders (VAEs) and its many variants (e.g. β-VAE [69], ACN [57] and others) that are widely used for SRL. Previous work has shown that VAEs can learn continually using generated samples from previous tasks, a method called Generative Replay [95]. S-TRIGGER uses Generative Replay to **remember information relative to previously encountered environments**. With the goal of an autonomous agent in mind, we complement our approach with a general method for **automatic detection of environment change**, based on the reconstruction error distribution of VAEs. This detection method allows the VAE to self-trigger the Generative Re-

play when a new environment is encountered, without the need for the user to specify environment changes. Finally, our approach respects important Continual Learning (CL) desiderata: no access to past data and bounded system size. Fig.3.5 presents an overview of S-TRIGGER.

We focus on settings where the agent has only access to sensory states (here, vision) that contain partial information about the environment, similarly to an embodied agent. The agent should be able to use learned knowledge to facilitate policy learning with RL. Hence, we test our approach on a 2-D first-person simulator with coherent physics, in two scenarii where the environment changes over time. The first scenario is a proof of concept with 2 environments, with minimal change between the two. We then test the robustness of our approach in a more challenging setting: sequences of randomly generated mazes. We test whether the learned features provide efficient and high-performing RL training on a navigation task. We also measure the ability of our method to retain past knowledge, using reconstruction error and visualization. Our results show that our approach avoids catastrophic forgetting, provides features that enable efficient RL and detects environment changes almost perfectly.

3.2.3 Related work

We review previous work related to Continual State Representation Learning for RL.

Generative models, e.g. VAEs, are widely used for SRL. Hence, one approach for Continual SRL is to apply a CL strategy developed for generative models. While most work in CL is focused on discriminative models, there is a recent interest in CL approaches for generative models [1, 118]. Proposed methods often rely on using generated samples to avoid forgetting [175, 95]. This technique, which we use in this contribution, is termed Generative Replay. For instance it is used in the algorithm VASE [1], where the authors propose a representation model that can, similarly to S-TRIGGER, learn continually and detect changes in the data distribution. VASE dynamically allows spare representation capacity to account for newly acquired information. While their approach is promising, the learned features were not tested on a RL setting, for which our approach is specifically designed and tested. Previous work [136] has demonstrated that learning state representation that performs well for policy learning is not straightforward. In our experiments, we demonstrate that S-TRIGGER can indeed continually learn features that allow fast and high-performing policy learning.

Other types of continual learning strategies for generative models could be investigated. We decided to select Generative Replay because it respects important desiderata for CL: bounded system size, no access to previous data. We found no previous work on applying these CL approaches to State Representation Learning for RL.

Another line of work on Continual SRL for RL is presented in DARLA [70], which circumvents the problem of catastrophic forgetting by learning disentangled representations with a specific VAE architecture. Their approach learns factored features that are robust to minimal modifications of the factors of the environment, ours continually update features as environment changes are detected. They show that learning a policy in a source domain with their representation as input can lead to zero-shot transfer by using the same policy in a target domain (similar to the source domain). The idea of learning one state representation model that generalizes to environments similar to previously encountered ones is important and compelling, but it is not clear how this model could handle heavy environment changes such as new objects appearing. In our experiments we shed light on the failure cases of this approach. However, both approaches are compatible so any progress on one side complements the other.

3.2.4 Continual State Representation Learning with Self-Triggered Generative Replay

We now present our method for Continual State Representation Learning, S-TRIGGER, which is designed for settings where the environment changes discretely over time.

Learning continually with Generative Replay

The considered problem is twofold: *build a State Representation model* which can *continually learn*, i.e. without forgetting. We use VAEs' encoding property to handle the State Representation and the generative ability of VAEs to generate states of previously seen environments to avoid forgetting, a technique known as Generative Replay [156]. In our case, we encode the sensory state received by the agent.

Once an environment change is detected, we generate states using the latent space of the VAE trained on previous environments, and add those generated samples to states collected in the new environment. Then, we train the same VAE on the joint data. This procedure is illustrated in Fig.3.5. The environment changes are detected automatically, as described thereafter.

This option does not re-use past data. Additionally, it is more scalable than a rehearsal method where samples from each previous environment would be maintained in memory, which amounts to scaling linearly in memory cost. Indeed, as new environments are encountered, we only need to maintain one VAE for all previously encountered environments, which makes Generative Replay a bounded system size solution in terms of memory cost.

In theory, the training cost should scale linearly in number of environment changes. However, in our experiments, we use a fixed number n of

generated samples for each environment change, and we consider up to two environment changes (i.e. sequence of three environments). n is of the same order of magnitude as the number of collected samples in the environment.

Self-Trigger: Automatic environment change detection

Since we aim at constructing an autonomous agent, we complement the proposed method with automatic detection of changes in the environment. Once an environment change is detected, Generative Replay is triggered, as described previously, which is why we refer to our method as Self-TRIGgered GEnerative Replay (S-TRIGGER for short).

The detection method is based on statistical hypothesis testing on VAE reconstruction error distribution. The intuition is: *if the VAE reconstruction error distribution suddenly changes, then the environment must have changed.*. To formalize this intuition, we use Welch's t-test [173] to test the hypothesis $\mathcal{H}_0$ that two sets x_1, x_2 of randomly sampled VAE reconstruction errors have equal mean. We choose this test over the standard t-test because we do not have reasons to assume that the two samples' variance are equal. The statistic t is, under $\mathcal{H}_0$, distributed as a Student distribution with a number of degrees of freedom ν that can be approximated using the Welch-Satterthwaite equation:

$$t = \frac{x_1 - x_2}{\sqrt{(s_1^2 + s_2^2)/N}}, \quad \nu \approx \frac{(N-1)(s_1^2 + s_2^2)^2}{(s_1^4 + s_2^4)}, \tag{3.3}$$

with N being the number of samples (assumed equal for both sets), and s_1, s_2 being the empirical standard deviations of samples 1 and 2, respectively. The decision of the test is based on a chosen significance level of α for the test, by using the p-value. If the p-value is below α, it suggests that the observed data is sufficiently inconsistent with $\mathcal{H}_0$ that $\mathcal{H}_0$ may be rejected and thus an environment change is detected.

Using the p-value on a statistical test is preferable for decision over using a threshold-based method directly on the error distribution. The threshold method is based on an arbitrary scale which depends on the considered sequence of environments. On the contrary, the test is a more general approach based on statistical principles and thus agnostic to scales. The p-value provides an interpretable parameter for controlling the detection method's recall, which is not possible with the threshold method. Additionally, we choose to use VAE reconstruction error distribution over the actual state distribution because we are interested in changes of the environment that require the VAE to be updated. For instance, if we add an already existing obstacle to the environment, the state distribution shifts, while the VAE reconstruction error distribution does not change because the VAE needs not to be updated. The test is lightweight and can thus be continually performed.

3.2.5 Experimental setting

Our experimental environments are developed in Flatland [23] presented in section 2.2.2.

Methods. For the State Representation model, we experiment with CCI-VAE [19], the state-of-the-art VAE model for learning disentangled representations.

For our RL experiments, we learn to solve navigation tasks using features from the trained VAE. The policies are trained in an on-policy fashion. At each time-step t we receive the sensory state input o_t, pass it through the SRL model (i.e. the VAE) to obtain the state representation s_t, and feed it to a policy $\pi(s_t)$ optimized with an RL algorithm, as described in Fig.1.3. For the RL algorithm, we selected the Proximal Policy Optimization (PPO) algorithm [151] of SRL Toolbox [137], which in our case employs an actor-critic network that both predicts the value function and a policy. We selected this method as it is a state-of-the-art policy gradient method, commonly used and robust to hyperparameter configurations. Policies are trained on-policy for $2M$ timesteps. Results are averaged over 5 seeds, and we test the significance of our results with statistical hypothesis testing. To compare learning curves, we present smoothed (moving average) and normalized (by the maximum performance) mean reward curves.

Evaluation. First, we evaluate whether the VAE successfully learned to reconstruct states, generate realistic states using sampling in the latent space and avoid catastrophic forgetting. We use visualization to measure performance. We also use the Mean Squared Error (MSE) to evaluate reconstruction quality and catastrophic forgetting over time as the environment changes.

Then, we evaluate the VAE ability to provide a state representation that enables efficient RL training, by measuring the performance of an RL agent using VAE features as input. The aim of SRL here is to have features that enable efficient behaviour learning hence the need to evaluate with RL. We use several baselines for comparison with S-TRIGGER (policy inputs are features of a VAE trained with our approach): Fine-tuning (policy inputs are features of a VAE sequentially fine-tuned), Source Only (policy inputs are features of a VAE trained only on first environment) and Upperbound (policy inputs are features of a VAE trained on data from all environment at once).

3.2.6 Experiment 1: Proof of concept

In this experiment we test S-TRIGGER on a continual learning setting with minimal changes between two environments. This experiment allows us to easily visualize the failure mode of fine-tuning on this setting, and provides insights on how our method solves the presented issues.

Description of the environments

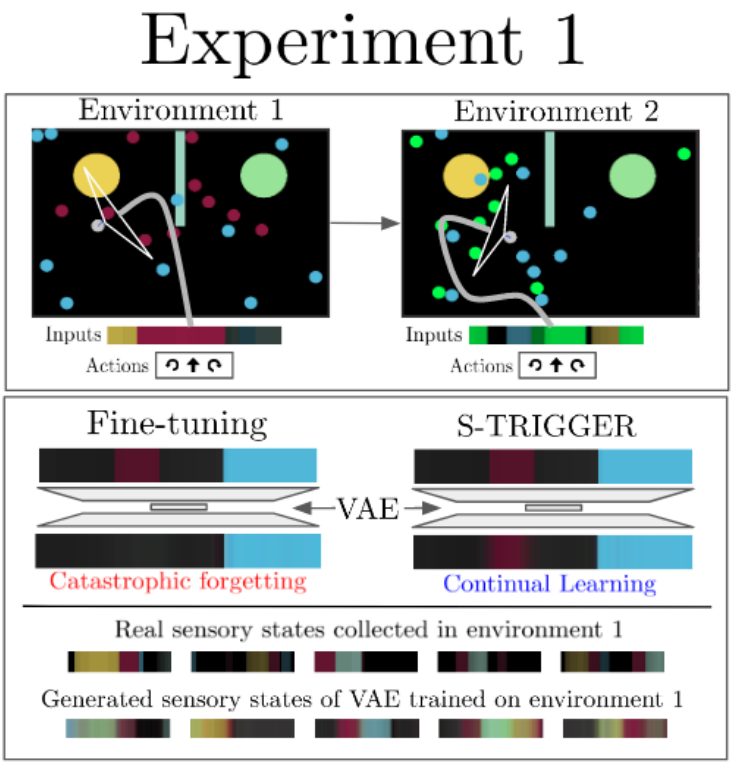


Figure 3.6: **Experiment 1**: *Top*: The two environments considered in Experiment 1. *Bottom*: Reconstruction comparison of sensory states from environment 1 between a VAE fine-tuned on environment 2 against a VAE trained with S-TRIGGER, followed by visualization of real and generated sensory states of a VAE trained in Environment 1.

We use two environments displayed in Fig.3.6. Most elements are identical between the two worlds: both are rooms of the same size with 3 fixed obstacles, 10 randomly placed round blue obstacles and 10 randomly placed round edible items. The only variation is the color of the edibles items. The environment changes once the representation model is finished training on the first environment. The navigation task consists in collecting as many edible items as possible in 500 timesteps. The input is the vision of the agent, i.e. a 1-D image corresponding to what the agent sees in front of it.

Results

Reconstruction evaluation We present in Fig.3.6 a qualitative evaluation of Fine-tuning and S-TRIGGER methods on VAE. Models are sequen-

tially trained on the first and then the second environment. Fine-tuning of VAE on the second environment leads to forgetting of the ability to reconstruct states from the first environment. On the contrary, S-TRIGGER successfully avoids this problem and the resulting VAE is able to properly reconstruct all elements of both environments, hence successful continual learning. Quantitatively, the MSE of reconstruction over 500 samples is similar for both methods on environment 2, whereas S-TRIGGER is one order of magnitude better than Fine-tuning on environment 1 (Table 3.1). This confirms our initial qualitative observations. We also present generated samples after the VAE is trained on the first environment in Fig.**??**. The VAE is able to generate realistic sensory states that are used by S-TRIGGER to remember past environments.

Table 3.1: Mean Squared Error (MSE) of reconstruction.

Strategy	MSE (environment 1)	MSE (environment 2)
Fine-tuning	$1.3 \cdot 10^{-3}$	$9.3 \cdot 10^{-4}$
S-TRIGGER (Ours)	$3.3 \cdot 10^{-4}$	$6.4 \cdot 10^{-4}$

Detection of environment change. Regarding environment change detection, our method tests whether a VAE trained on the first environment has a mean reconstruction error statistically different between states of environments 1 and 2. The reconstruction error mean is significantly higher on frames of environment 2, since the VAE has only been trained on frames from environment 1. We take advantage of this observation as we use a Welch's t-test to compare two batches of mean VAE reconstruction error, computed on randomly collected states from each environment. The null hypothesis is rejected if the p-value is greater than 0.01. We repeat this experiment 5000 times. The test is 100% successful when it should detect an environment change, and 99.5% successful when it should not detect a change. Any chosen critical p-value between 0.05 and 0.0001 provides similar results.

RL evaluation. Learning curves and final performances are presented in Fig.3.7. We can observe catastrophic forgetting with Fine-tuning, as its performance is significantly inferior to other models on Environment 1. Indeed, this is expected since we previously showed that the model has forgotten how to reconstruct states of Environment 1. We also observe that VAE's features support a form of zero-shot transfer. Indeed, using features of a VAE trained on the first environment only (see Source Only curve in Fig.3.7) to learn to solve Task 2 is remarkably efficient.

On Task 1, our method S-TRIGGER performs on par with a model trained only on Environment 1, which shows the absence of forgetting enabled by Generative Replay. On Task 2, the difference between models is less significant. However, S-TRIGGER still performs better or on par with Fine-Tuning.

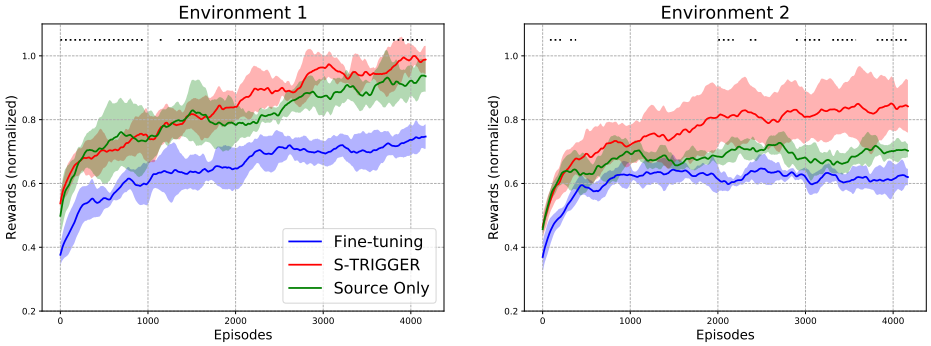


Figure 3.7: Smoothed normalized mean reward and standard error over 5 runs of RL evaluation using PPO with different inputs. Each method is described in Sec.3.2.5. Dots indicates significance when testing against Fine-tuning with a Welch's t-test at level $\alpha = 0.05$.

3.2.7 Experiment 2: Robustness tests

In this experiment, we test the robustness of S-TRIGGER on a scenario with sequences of randomly generated mazes. The setting is more challenging compared to Experiment 1 because of both the extended number of environments (3 sequential environments) and the heavier changes between them (changes of color, location, size of all elements).

Description of the environments

We use sequences of three randomly generated mazes. The mazes are composed of rooms and corridors, and randomly placed edibles: fruits (+10 reward) and poisons (−10 reward). Fruits share the same color, different from poisons. In the sequence, rooms and corridors change location, size and color. The size of edibles gradually increase in the sequence, and their color changes as well. The environment changes once the representation model is finished training. In each of the three environments, the RL task is to learn how to collect fruits while avoiding poisons. The setting is illustrated in Fig.3.8.

Results

We first evaluate the reconstruction error and the environment change detection on a set of 100 sequences of 3 randomly generated mazes. Then, we present the Reinforcement Learning evaluation on one sequence of 3 mazes.

Reconstruction evaluation. In this experiment, we compare S-TRIGGER and Fine-tuning by sequentially training a VAE on sequences of 3 randomly generated mazes, using randomly collected sensory states. We repeat the process 100 times with different mazes each time. As in the previ-

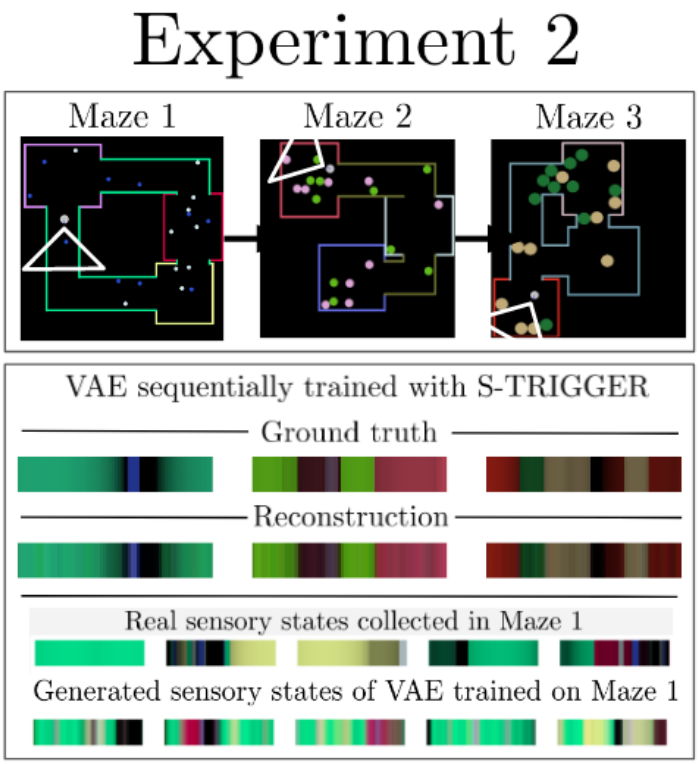


Figure 3.8: **Experiment 2**: *Top*: One of the sequences of 3 randomly generated mazes considered in Experiment 2. *Bottom*: Reconstruction of sensory states by a VAE sequentially trained with S-TRIGGER on the 3 mazes presented at the top, followed by visualization of real and generated sensory states of a VAE trained in Maze 1.

ous experiment, contrary to Fine-tuning, S-TRIGGER does not forget how to reconstruct sensory states, and performs one order of magnitude better on previously encountered environments, i.e. Mazes 1 and 2. Qualitatively, we can observe how S-TRIGGER avoids catastrophic forgetting in Fig.3.8: after encountering Maze 3, the VAE reconstruction is satisfactory on all 3 mazes, hence the state information has been successfully compressed and this information has been kept over time. Also, the generation quality is still satisfactory, as seen in Fig.3.8 (bottom).

Notice that the reconstruction is as satisfactory on Maze 1 as Maze 2 ($5.82 \cdot 10^{-3}$ vs. $5.24 \cdot 10^{-3}$), even though Maze 1 has been experienced by the agent more learning steps in the past than Maze 2. The VAE is still able to generate enough samples of Maze 1, and with a sufficient generation quality, to learn a satisfactory representation when Maze 3 is encountered. Also, since we obtained successful results on 100 different sequences of 3 mazes,

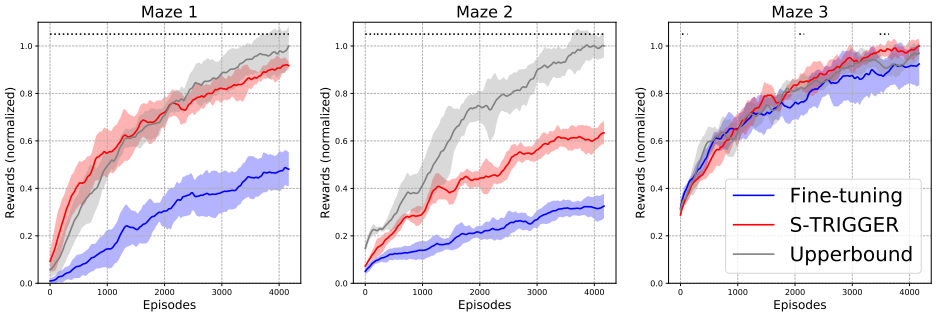


Figure 3.9: Randomly generated mazes: smoothed normalized mean reward and standard error over 5 runs of RL evaluation using PPO with VAE features as inputs. Each method is described in Sec.3.2.5. Dots indicate significance when testing against Fine-tuning with a Welch's t-test at level $\alpha = 0.05$.

we can infer that S-TRIGGER is successful on any sequence of 3 randomly generated mazes. Our method is thus robust to the considered continual learning scenario.

Detection of environment change. As for the environment change detection method of S-TRIGGER, we test it on the 100 sequences, by artificially creating 400 transitions between environments per sequence (200 with a change, 200 without a change), which sums up to 40000 different transitions. The method performs quasi-perfectly well, with at least 99% precision and recall. Considering the fact that it can be ran continuously for a low computational cost, the method is well suited for the considered setting.

RL evaluation. For experiments in this section, we randomly selected one sequence on 3 randomly generated mazes, and kept it fixed for all experiments. VAE and policies are all trained on this same sequence of mazes, presented in Fig.3.8. We compare performance of policies trained on-policy with the RL algorithm PPO. Policies are trained using VAE features as inputs. VAEs are trained with three different strategies: Fine-tuning, S-TRIGGER (both sequentially trained) and Upperbound (jointly trained on data from the three mazes, not sequential). Learning curves and final performances are presented in Fig.3.9.

S-TRIGGER performs at least twice better than Fine-tuning on Maze 1 and 2, and performs optimally w.r.t Upperbound on Maze 1 and 3. Indeed, Upperbound has access to all data at once, which makes it an optimal solution w.r.t to using VAE features for policy learning in the considered Continual learning scenario. While S-TRIGGER cannot always reach the performance of this Upperbound (it does on Maze 1, but not on Maze 2), it clearly improves Continual SRL: the learned representation model allows fast and more efficient policy learning with a standard RL algorithm by

limiting forgetting of previously seen environments.

Compared to Experiment 1, Fine-tuning performs worse on previously seen environments, due to the increased difference between encountered environments. Since the size of edibles increase for each new maze, a VAE is sequentially trained with Fine-tuning until Maze 3 is not trained to see small objects in Maze 1, which makes the agent almost entirely *blind* in Maze 1. On the contrary, S-TRIGGER allows the VAE to see objects in all mazes, which contributes to having faster and better-performing learned policies. Fine-tuning has one of the expected shortcomings of approaches like DARLA [70]. When there is too much change between environments, learning a sufficiently general representation that can transfer to all environments is not feasible. The representation model needs to be updated when a new environment is encountered.

3.2.8 Conclusion

We described S-TRIGGER, a Continual State Representation Learning method capable of learning as the environment changes discreetly. Our method automatically detects changes and relies on using generated samples of previous environments, with a fixed-size system. It learns a unique representation model that compresses information of each encountered environment, which can be used to train policies with RL, efficiently and with high-performance.

3.3 Contribution: DisCoRL

3.3.1 Abstract

In multi-task reinforcement learning there are two main challenges: at training time, the ability to learn different policies with a single model; at test time, inferring which of those policies applying without an external signal. In the case of continual reinforcement learning a third challenge arises: learning tasks sequentially without forgetting the previous ones. In this section, we tackle these challenges by proposing DisCoRL, an approach combining state representation learning and policy distillation. We experiment on a sequence of three simulated 2D navigation tasks with a 3 wheel omnidirectional robot. Moreover, we tested our approach's robustness by transferring the final policy into a real life setting. The policy can solve all tasks and automatically infer which one to run, as shown in the supplementary video: `https://youtu.be/mzUigGWEfbU`.

3.3.2 Introduction and contribution

In this contribution, we propose to address a continual learning problem of reinforcement learning. In this continual learning setting, each learning

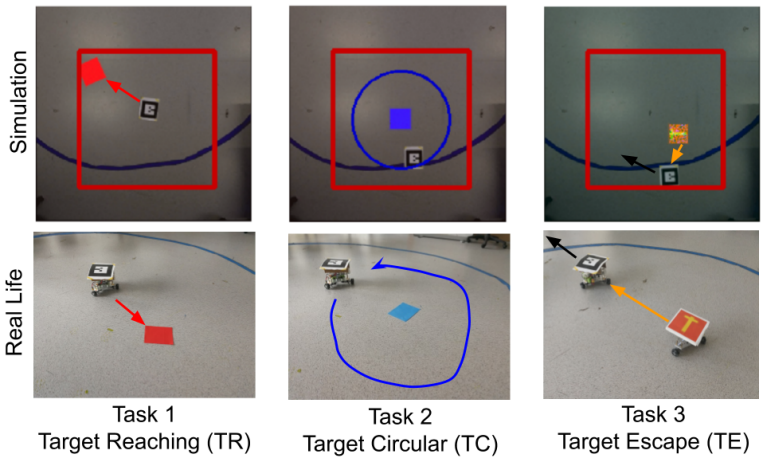


Figure 3.10: Image of the three tasks, in simulation (top) and in real life (bottom) sequentially experienced. Learning is performed in simulation, real life is only used at test time.

experience is called a task and a task solution is a policy. Our goal is to propose a learning setting compatible with a real autonomous agent. To this end, we propose three simulated robotics tasks and propose an approach that will solve such tasks sequentially. At each task, the agent should learn a policy based on a reward function and a RL algorithm.

In order to fit reinforcement learning into a continual setting, we use a method called policy distillation [145] that allows the transfer of several policies learned sequentially into a single model. To validate our approach, we evaluate the final results on the three simulated learning settings but also in a real life setting similar to the simulation (Figure 3.10). It is important to note that, at test time, the agent does not have access to a task label to determine which policy to run, and thus, it needs to figure it out by itself from its observations.

Our contribution is to propose **DisCoRL** (*Distillation for Continual Reinforcement learning*): a modular, effective and scalable pipeline for continual RL. This pipeline uses policy distillation for learning without forgetting, without access to previous environments, and without task labels. Our results show that the method is efficient and learns policies transferable into real life scenarios.

3.3.3 Related work

Multi-task RL: The objective of Multi-task learning (MTL) [22] is to learn several tasks simultaneously; generally by training tasks in parallel with an unique model. Therefore, multi-task RL aims at constructing one single policy that can solve a number of different tasks. Note how in classi-

fication this problem is quite simple, as data from all tasks just have to be shuffled randomly and can then be learned all together at once. However, in RL environments, data is sampled on sequences that can not be shuffled randomly with all other environments because the environments are not accessible simultaneously. Learning multiple tasks at once is thus more complicated.

Policy distillation [145] can be used to merge different policies into one single module/network. This approach uses two models, a trained policy (the teachers) to annotate data with soft-annotations, and a model to learn from the former (the student). The student is trained in a supervised manner with the soft-labels. The soft-annotation is supposed to help the student to learn faster than the teacher did [51]. Policy distillation can be used then to learn several policies separately and simultaneously, and distill them into a single model as in the distral algorithm [164]. In our approach, we also use distillation but we do not keep the teacher model, we just label a set of data and then delete the teacher. Furthermore, tasks are learned sequentially, and not simultaneously. Other approaches such as SAC-X [142] or HER [2] take advantage of Multi-task RL by learning auxiliary tasks in order to help learning a main task. This approach is extended in the CURIOUS algorithm [33]. It selects tasks to be learned that improve an absolute learning progress metric the most.

Continual Learning in RL: In the context of continual RL, several approaches have been proposed, such as the use of *Progressive Nets* in [146], *Elastic Weight Consolidation (EWC)* [85], *Progress And Compress* (P&C) [153], or *CRL-Unsup* [106]. However they either need a task indicator at test time to choose which policies to run or, they have some hyper-parameter difficult to tune during a continual learning training, such as the importance of the Fisher information matrix in EWC. Our method does not add any new hyper-parameter to tune during the sequence of tasks and does not need a task label at test time.

RL in Robotics: Applying RL to real-life scenarios such as robotics is a major challenge that has been studied widely. One of the major problems in this setting is that sampling data and fortiori learning is costly. Therefore sample efficiency and stability in learning are highly valuable. One common approach to reduce training cost is training policies in simulation and then deploying them in real-life, hoping that they will successfully transfer, considering the gap in complexity between simulation and the real world. Such approaches are termed *Sim2Real* [55], and have been successfully applied [29, 111] in many scenarios. One of these approaches is Domain Randomization [166], which we use in this contribution. This technique trains policies in numerous simulations that are randomly different from each other (different background, colors, etc.). Using this technique, the transfer to real life is easier.

Others have tried to train a policy directly on real robots, facing the

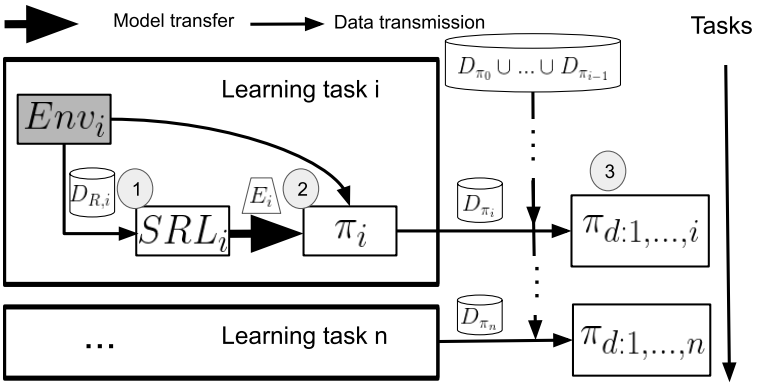


Figure 3.11: Overview of our full pipeline for Continual Reinforcement Learning. White cylinders are for datasets, gray squares for environments, and white squares for learning algorithms, whose name corresponds to the model trained. Each task i is learned sequentially and independently by first generating a dataset $D_{R,i}$ with a random policy to train a state representation with an encoder E_i with an SRL method (1), then we use E_i and the environment to learn a policy π_i in the state space (2). Once trained, π_i is used to create a distillation dataset D_{π_i} that acts as a memory of the learned behaviour. All policies are finally compressed into a single policy $\pi_{d:1,..,i}$ by merging the current dataset D_{π_i} with datasets from previous tasks $D_{\pi_1} \cup ... \cup D_{\pi_{i-1}}$ and using distillation (3).

hurdle of the lack of sample efficiency of RL algorithms. SAC-X [142] is one example that takes advantage of multi-task learning to improve efficiency, by simultaneously learning the policy and a set of auxiliary tasks to explore its observation space - in search for sparse rewards of the externally defined target task.

In the literature, most approaches focus on the single-task or simultaneous multi-task scenario. In this contribution, we attempt to train a policy on several tasks sequentially and deploy it in real life by combining policy distillation, training in simulation and state representation learning.

3.3.4 Methods

In this section we present our approach towards continual reinforcement learning for a sequence of vision based tasks. We assume that observations visually allow us to recognize the current task from other tasks. We first explain how we learn a single task by combining state representation learning (SRL) and reinforcement learning (RL), then how each task is incorporated in the continual learning pipeline. Finally, we present how we evaluate the full pipeline.

Learning one task

Each task i is solved by first learning a state representation encoder E_i in order to compress input images into a representation of the important underlying factor of variation. This step reduces the input space for the reinforcement learning algorithm and makes it learn more efficiently [136]. To train this encoder, as shown in Fig. 3.11 (left), we sample data from the environment Env_i with a random policy. We call this dataset $D_{R,i}$. $D_{R,i}$ is then used to train the SRL model composed of an *inverse model* and an *auto-encoder*. The inverse model is trained to predict the action a_t that led to transition from state s_t to s_{t+1}, both extracted from respective observations o_t and o_{t+1} by the auto-encoder using E_i. The auto-encoder is additionally trained to reconstruct the observations from the encoded states. The architecture is motivated by the results from [136].

Once the SRL model is trained, we use its encoder E_i to provide features as input to a policy π_i trained using RL. We also experimented to learn the policy directly in the raw pixel space but, as shown in [136], it was much less sample efficient.

Once π_i is learned, we use it to generate sequences of on-policy observations with associated actions, which will eventually be used for distillation (Fig. 3.11, right). We call this the distillation dataset D_{π_i}. We generate D_{π_i} the following way: we randomly sample a starting position and then let the agent generate a trajectory. At each step we save both the observation and associated action probabilities. We collect the shortest sequences maximizing the reward for an episode.

We also experiment to generate D_{π_i} with a regular sampling and a random policy to generate the states that we annotate with π_i to compare results, as detailed in section 3.3.6 below.

From each task we only keep the dataset D_{π_i}. As soon as we change the task, $D_{R,i}$ and Env_i are not available anymore. D_{π_i} is split into a training set and a validation set.

Learning continually

To learn continually, we adapt policy distillation [145] to a continual learning setting. The distillation consists of training a student policy to imitate a teacher policy. In our case, a student model learns from a teacher policy the action probability associated with each observation. Each dataset D_{π_i} allows to distill the policy π_i (the **teacher** model) into a new network $\pi_{d:i}$ (the **student** model). In classic distillation, both data and models need to be saved, however saving just soft-annotated data is a lighter solution adapted to a continual setting.

With the aggregation of several distillation datasets D_{π_i}, we can distill several policies into the same network that can achieve all tasks (Fig.3.11,

bottom right). By extension of the previous nomenclature, we denote $\pi_{d:1,..,n}$ a model where policies π_1 ... π_n have been distilled in. When distilling all policies into the student, we select our best models with early stopping, and test later in simulation and in real life settings.

Since we assume that observations visually allow us to recognize the current task, $\pi_{d:1,..,n}$ is able to choose the right action for the current task without a task indicator.

The method, termed *DisCoRL* for *Distillation for Continual Reinforcement learning*, allows us to continually learn several policies while minimizing forgetting. Regarding scalability, saving data from all past experiments may not look ideal if there is a high number of tasks. However, this solution is highly effective for remembering and letting the reinforcement learning algorithm be absolutely free to learn a new policy without regularization. It is worth mentioning that RL is the real bottleneck in the whole process: Dataset D_{π_i} contains approximately 10k samples per task, which allows to perform the distillation quickly, relative to how long and computationally expensive RL is (few minutes needed to learn $\pi_{d:i}$ while several hours are needed to learn π_i). Thus, in this context, it is better not to curb RL with regularization. Indeed, as explained in Section 3.3.6, we tried several regularization based approaches that were not successful.

Evaluation

The first evaluation is the performance of the final policy on the simulated environment. This evaluation can then be compared with the performance of each teacher policy. For the second evaluation we test if the policy is robust to the reality gap and can be adapted into a real life scenario. The simulation is voluntarily close to the real life setting but the reality gap is notoriously problematic.

To get an insight on the evolution of the distilled model, we also save distillation datasets at different checkpoints while learning each task. By distilling and evaluating at several time steps, we assess catastrophic forgetting on previous tasks when fine tuning on new tasks.

3.3.5 Experimental setup

We apply our approach to learn continually three 2D navigation tasks applicable in real life. The software related to our experimental setting is available online[1].

[1] `https://github.com/anonymous-authors-2018/CoRL`

Robotic setup

The experiments consisted of 2D navigation tasks using a 3 wheel omni-directional robot similar to the 2D mobile navigation in [137]. The input image is a top-down view of the floor and the robot is identified by a black QR code. The room where the real-life robotic experiments are performed is lit by surrounding windows and artificial illumination and is subject to illumination changes depending on the weather and time of the day. The robot uses 4 high level discrete actions (move left/right, move up/down in a cartesian plane relative to the robot) rather than motor commands.

We simulate the experiment to increase sampling and learning speed. The simulation is performed by artificially moving the robot picture inside the background image according to the chosen actions. We use domain randomization [166] to improve the stability and facilitate transfer to the real world : during RL training, at each timestep, the color of the background is randomly changed.

Continual learning setup

Our continual learning scenario is composed of three similar environments, where the robot is rewarded according to the associated task (Fig. 3.10). In all environments, the robot is free to navigate for up to 250 steps, performing only discrete actions within the boundaries identified by a red line. Each task is associated with a visual target, which color depends on the task. This way, the controller can automatically infer which policy it needs to run and thus, does not need task labels at test time.

Task 1. The task of environment 1 is named Target Reaching (TR). The robot gets at each timestep t a positive reward $+1$ for reaching the target (red square), a negative reward -1 for bumping into the boundaries, and no reward otherwise.

Task 2. The task of environment 2 is named Target Circling (TC). The robot gets at each timestep t a reward R_t defined in Eq. 3.4 (where z_t is the 2D coordinate position with respect to the center of the circle) designed for agents to learn the task of circling around a central blue tag. This reward is highest when the agent is both in the circle (red (first) part in Eq. 3.4), and has been moving for the previous k steps (blue, second part). An additional penalty term of -1 is added to the reward function in case of bumping the boundaries (last, green part). A coefficient $\lambda = 10$ is introduced to balance the behaviour.

$$R_t = \lambda * \boxed{(1 - \lambda(\|z_t\| - r_{circle})^2)} * \boxed{\|z_t - z_{t-k}\|_2^2} + \lambda^2 * \boxed{R_{t,bump}} \tag{3.4}$$

Task 3. The task of environment 3 is named Target Escaping (TE). Robot A is being chased down by another robot B with an orange tag. Robot B is hard-coded to follow robot A, and robot A has to learn to escape using RL. Robot A gets at each timestep t a reward of $+1$ if it's far enough from robot B, otherwise, if it is in the range of robot B, it gets a reward of -1. Additionally, robot A gets a negative reward of -1 for bumping into the boundaries.

All RL tasks are learned with PPO [152] and the same state representation learning (SRL) model, as described in section 3.3.4. We select the model architecture as in [137] for RL and SRL. The input observations of all models are RGB images of size $224 * 224 * 3$.

3.3.6 Results

We first present our design choices for the distillation process: loss functions and data sampling strategies. We then use these choices to present our main result: the distillation of three tasks continually into a single policy that can achieve the three tasks both in simulation and real-life. We provide a supplementary video[2] of this policy deployed in real-life on the robot showing the successful behaviors. We also present the different strategies we tried but that did not work in our setting.

Evaluation of distillation

Distillation strategies: Distillation is done with a loss function that minimizes the difference between the student model's output and the teacher model's output for the same input. As in the policy distilation paper [145], we investigate variations of the loss function : Mean Squared Error loss

$$\mathcal{L}_{MSE}(x, y) = \mathbb{E}[||x - y||_2^2]$$

Kullback-Lieber divergence, and Kullback-Lieber divergence with temperature smoothing

$$\mathcal{L}_{KL,\tau}(p|q) = \mathbb{E}[\text{softmax}(\frac{p}{\tau})ln(\frac{\text{softmax}(\frac{p}{\tau})}{\text{softmax}(q)}))]$$

We run a performance comparison of the different losses by computing the mean normalized performance of a student policy trained to perform all three tasks (Tab.3.2). Using the Kullback-Lieber divergence loss function with temperature smoothing with $\tau = 0.01$ performed the best, and optimizing the temperature parameter yields a small performance boost. This result is coherent with [145] where they reach the same conclusion.

[2] `https://youtu.be/mzUigGWEfbU`

Distillation loss	Student performance (± std)
MSE	0.71 (± 0.22)
KL ($\tau = 1$)	0.76 (± 0.14)
KL ($\tau = 0.1$)	0.68 (± 0.18)
KL ($\tau = 0.01$)	**0.77 (± 0.13)**

Table 3.2: Mean normalized performance.

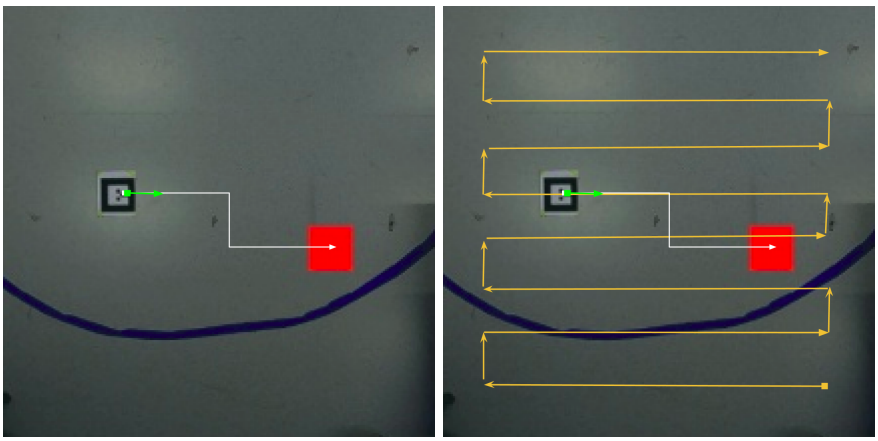


Figure 3.12: Representation of data generation strategies to distill the teacher policy. Left, on policy sampling. Right, grid sampling.

Data sampling strategies: We evaluate the effect of three different sampling strategies to create D_{π_i} for policy distillation. Data sampling is a key component as the sampled dataset should be as small as possible but contain sufficient information for student model training. The strategies involved for data generation are:

- *On-policy generation (Fig.3.12, left)*: We start an episode from a random point, then at each timestep t, we collect an observation o_t and perform the action $a_{\pi_i,t}$ of the teacher policy.

- *Off-policy generation from a grid walker (Fig.3.12, right)*: at each timestep t, we collect an observation o_t by performing an action $a_{grid,t}$ of a *grid walker* exhaustively exploring the space of the arena. However, for each o_t we save the probability of action $p(a_{\pi_i,t} \mid o_t)$ that would have been taken by the teacher policy. The goal of this strategy is to provide a more exhaustive sampling of the space of robot positions.

- *Off-policy generation from a random walker*: similar to the previous strategy, but with an untrained policy, i.e. from a policy with random weights with input in the raw pixels' space, instead of a grid walker.

D_{π_i} is thus composed of tuples $(o_t, p(a_{\pi_i,t} \mid o_t))$, with $p(a_{\pi_i,t} \mid o_t)$ the action probability associated to the action $a_{\pi_i,t}$ taken by the teacher, i.e., a *soft label*, since we use the Kullback-Lieber divergence loss.

Performance of policies distilled using such strategies (see Fig. 3.13) show that *on-policy generation* (i.e., demonstrations) suffice to reproduce performance close to those of teacher policies on every task individually, with reasonable stability. In particular cases, see Fig. 3.13 for task TC, this strategy even provides a small boost in performance in the student policy over the teacher policy.

On the contrary, using *off-policy data generation from a grid walker* for distillation results in either unstable or poorly performing policies, especially in tasks defined by a reward function requiring the agent to move actively (*TC* task, blue part of eq. 3.4) or anticipate the behaviour of another agent (*TE* task). In this case, the resulting policy reaches the performance of a lower-bound baseline obtained by distilling from trajectories of an untrained policy (see *Student on off-policy data with a random walker* in fig. 3.13), i.e. from a policy with random weights with input in the raw pixels' space.

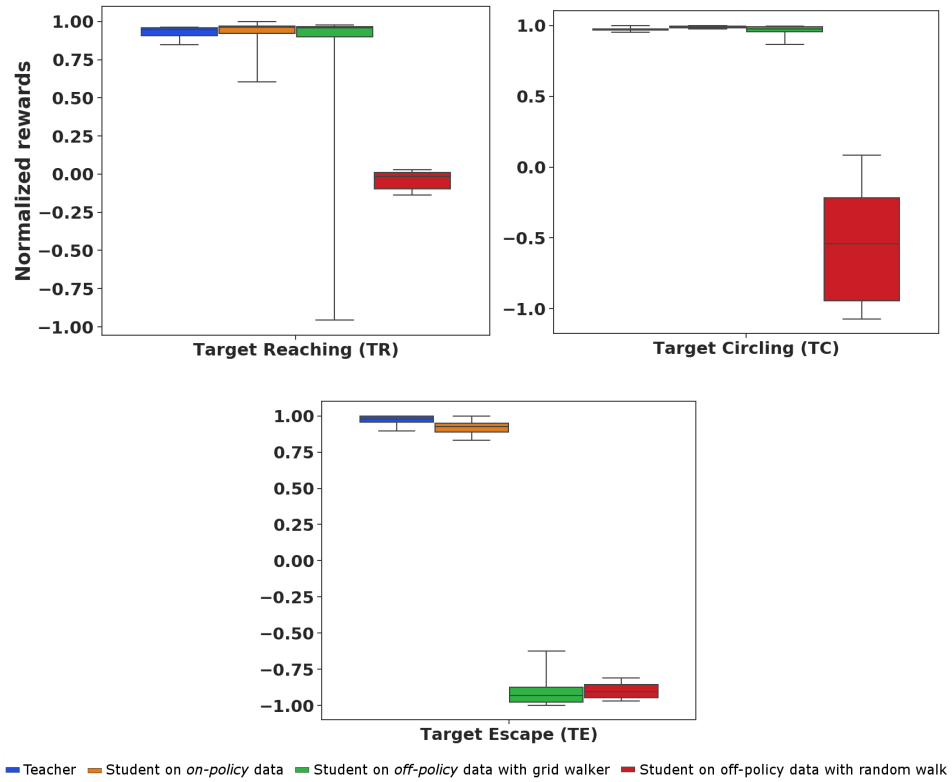


Figure 3.13: Efficiency (normalized rewards w.r.t the best teacher performance) of policies distilled on 8 seeds using various data generation strategies for each task separately. Each evaluated policy is distilled on 15k tuples of sampled observations and action probabilities, for 4 epochs.

Main result

We present our final results in Fig. 3.14. We used *on policy* data generation and training using KL divergence loss with $\tau = 0.01$. We show box plots over 10 episodes of reward performances for teacher policies in each task, and for the distillation of the same three teachers into a single student using DisCoRL. Each policy is evaluated in simulation and also in real-life on the robot. As a reference, we also show the performance of a random agent in each task. Our approach is effective in a continual reinforcement learning setting: the performance of teachers and students are similar.

More precisely, there are two main challenges to overcome in our setting: learning a behaviour via distillation by using only a limited number of examples, and the reality gap which can notoriously [166] introduce variations that may lead the policy to fail. Fig. 3.14 demonstrates the efficiency of our approach at overcoming both of these issues: only a small fraction of performance is lost from teacher to student, and from simulation to reality. We can see that the single student distilled policy achieves close to maximum rewards in all tasks, in real-life.

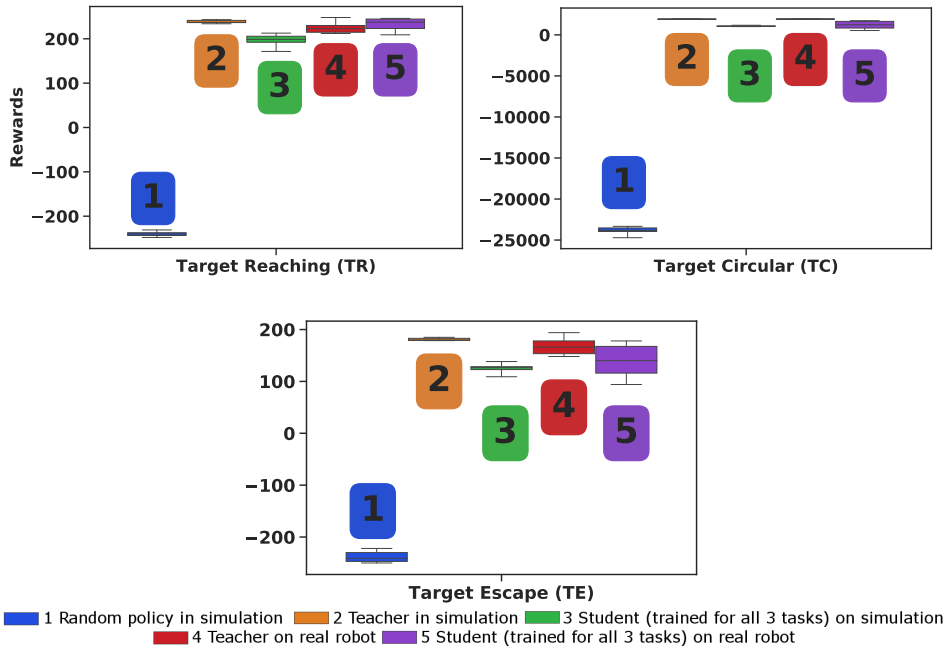


Figure 3.14: Main result: distillation in a continual learning setting of three teacher policies into a single student policy. The resulting policy is able to perform all three tasks both in simulation and in the real world, while minimizing forgetting.

Negative results

While distillation is effective for policy transfer, we also tested other alternatives worth mentioning.

Elastic Weight Consolidation (EWC) [85] was implemented as a continual learning baseline to compare with the distillation method. EWC has the appealing advantage of not re-using any data from previous tasks. However, in all cases we found the method unsuccessful.

Tuning the λ parameter that controls the trade-off between weight protection and learning the new task showed that either λ is too low and catastrophic forgetting happens, or λ is too high and nothing new is learned (i.e., the full network is frozen). A λ value providing a proper balance in between both effects could not be found for such sequential tasks to be learned.

Progress and Compress (P&C) [153] was tested but as EWC, we had problems with the importance factor λ and we were not able to learn three policies into a single model with this method.

Adding task labels for distillation. Even if all tasks contain a visually differentiating identifier, they remain visually similar. In cases, we found that a distilled policy trained to perform well on several tasks can mix up tasks and thus not perform adequately. Hence, either adding tasks labels directly, or adding a module in the network that predicts the task label could be a way to improve the efficiency of distillation. However, none of the approaches were successful in practice, yielding the same results with or without task labels.

3.3.7 Discussion

Continual learning is a complex field: every setting is different and expectations may vary from one algorithm to another. For example it is not easy to compare results with and without a task indicator. Task labels always add information to learn or test, and thus, they often improve results. However in a realistic setting they may be lacking.

Otherwise, the scalability vs stability trade-off is a difficult question. Learning online in a single model or a dual architecture scales well to a high number of tasks. However, this solution is often unstable, in particular because if a task fails, there is a high risk of forgetting everything that has been learned previously. For example, in generative replay, the generator is used as a memory. However, if at some moment it diverges while learning, all data from the past is destroyed.

The approach we propose uses soft-labelled samples as a memory, similarly to rehearsal methods, which will grow the memory continually. Compared to S-TRIGGER, where the generative model acts as memory and thus allows to keep a bounded system size, DisCoRL is less memory-efficient. However, rehearsal brings no risk of forgetting or destroying past knowl-

edge. With generative models, compounding errors might accumulate, and we might lose accuracy as more and more environments and tasks are presented.

Nevertheless, even if we believe this work proposes a stable and scalable framework for continual reinforcement learning, several possibilities for improvement exist. Future work includes having not only a policy learned in a continual way, but also the SRL model associated. We would need to update the SRL model as new tasks are presented sequentially. One possible approach would be to use Continual SRL methods like S-TRIGGER [25] or VASE [1]. Moreover, we would like to optimize more the memory needed to save samples by reducing their number and their size.

Finally, training policies on real robot experiences without the use of simulation would be desirable. However, at the moment, this is more a RL challenge than a CL challenge. One promising approach would be to use model-based RL while learning the state representation learning (SRL) model to improve sample efficiency. Though nowadays approaches still do not offer solutions that work in a reasonable amount of time.

3.3.8 Conclusion

In this contribution we presented DisCoRL, an approach for continual reinforcement learning. The method consists of summarizing sequentially learned policies into a dataset to distill them into a student model. It allows to learn sequential tasks in a stable pipeline without forgetting. Some loss in performance may occur while transferring knowledge from teacher to student, or while transferring a policy from simulation to real life. Nevertheless, our experiments show promising results in simulated environments and real life settings.

3.4 Conclusion on Representation Learning

In this chapter, we proposed several approaches to the learning of perception using various techniques from the machine learning community. We showed in particular that SRL and RL can be extended to Continual Learning setups where the environments, or the tasks change and the agent has to remember past events and scenes, just like what an agent in a human environment would have to do.

However, these approaches still suffer from many limitations for our goal of learning a perception similar to humans. For example, reinforcement learning is not trained for general understanding of the environment and perception, but for finding a viable solution that maximizes the accumulated reward on a particular task. In most cases, the algorithm learns a representation of data which is not general: if the environment changes slightly, the policy will not work anymore for maximizing reward. Indeed,

the learned representation, and basically what the policy has learned, is not general. Thus, what is learned during RL cannot be easily re-used for other tasks. Concerning Continual Learning, the literature still mostly focuses on static datasets like MNIST. In our contributions, we tried to go beyond this static dataset setup but this progress is hard because there are no benchmarks in the RL setup. This is because there are numerous possible tasks in the continual learning setup for reinforcement learning. In the environment, objects could change, or the background, or additional objects can appear, or new agents can appear. Otherwise the agent can change as well, in terms of sensors, or in terms of motors, or both. All these possibilities need to be summarized and organized in a common widely accepted benchmark for progress to be made. Until then, much progress has to be made before CL can really deliver on its promises: a never-ending learning agent that learns continually just like humans do.

Finally, there is the additional issue of disentanglement with all the previously mentioned approaches. Disentanglement is the study of the organization of latent spaces. All the previous approaches learn representations of data in a vectorial format, but the organization of this latent space is crucial for using it on downstream tasks. Essentially, if we understand the structure of the latent space, it is intuitively easier to manipulate it and re-purpose it. A disentangled latent space would isolate high-level features of data on one or few dimensions of the latent space, allowing easy manipulation of data and interpretability, which could in turn help solve new tasks that might be harder with entangled latent spaces. We investigate this question of disentanglement for latent spaces in the next Chapter.

Chapter 4

The role of Actions in Disentangled Representation Learning

Contents

4.1 On the importance of disentanglement

How can intelligent agents solve a diverse set of tasks in a data-efficient manner? The disentangled representation learning approach posits that such an agent would benefit from separating out (disentangling) the underlying

structure of the world into disjoint parts of its representation, reflecting the compositional nature of the world. Previous work [70, 136] in this field of research has shown that agents capable of learning disentangled representations can perform more data-efficient policy learning. More generally, disentangling the different elements of a scene is an intuitive ability of humans and animals. Intelligent agents isolate the factors of variations of the world in order to manipulate them for achieving tasks and goals. For a software agent, this skill does not emerge automatically, hence the need for algorithms that create disentangled understanding of the world. In usual representation learning without disentanglement, the algorithm will just learn a sufficient representation to solve the particular task it is asked to learn, and will not purposely create a re-usable representation that generalizes to many tasks.

However, there is no generally accepted formal definition of disentanglement in Representation Learning, which prevents significant progress in this emerging field. Recent efforts have been made towards finding a proper definition [105]. In this thesis we focus on the definition proposed by Higgins et al. [68]: Symmetry-based Disentangled Representation Learning. We extend their theory and provide practical guidelines for implementation.

4.2 Symmetry-based Disentangled Representation Learning

Higgins et al. [68] define Symmetry-Based Disentangled Representation Learning (SBDRL), by taking inspiration from the successful study of symmetry transformations in Physics. Their definition focuses on the transformation properties of the world. They argue that transformations that change only some properties of the underlying world state, while leaving all other properties invariant, are what gives exploitable structure to any kind of data. For instance, a ball might be moving in many directions but will not usually change shape often, and will even less often change colors.

They distinguish between linear and non-linear disentangled representations, which models whether the transformation affects the representation in a linear or non-linear way. Supposedly, linearity might be more useful for downstream tasks such as Reinforcement Learning or auxiliary prediction tasks, since they model the action of the transformations on the representation in a simpler and interpretable way that what might be learned in non-linear way. Their definition is intuitive and provides principled resolutions to several points of contention regarding what disentanglement is. For clarity, we refer to a representation as SB-disentangled if it is disentangled in the sense of SBDRL, and as LSB-disentangled if linearly disentangled.

Mathematical formalization

The core idea is that SB-disentanglement of a representation is defined with respect to a particular decomposition of the symmetries of the environment. Symmetries are transformations of the environment that leave some aspects of it unchanged. For instance, for an agent on a plane, translations of the agent on the y-axis leave its x coordinate unchanged. They formalize this using group theory. Groups are composed of these transformations, and group actions are the effect of the transformations on the state of the world and representation. We now recall the formal definition of a SB-disentangled representation w.r.t to this group decomposition. We advise the reader to refer to the detailed work of [68] for any clarification.

The proposed definition of SB-disentanglement supposes that the symmetries of the environment are formally defined as a group G (equipped with composition) that can be decomposed into a direct product $G = G_1 \times .. \times G_n$. Let W be a set of world-states $W = (w_1, .., w_m) \in \mathbb{R}^{m \times d}$, where each state w_i is a d-dimensional vector. We suppose that there is a generative process $b : W \to O$ leading from world-states to observations (these could be pixel, retinal, or any other potentially multi-sensory observations), and an inference process $h : O \to Z$ leading from observations to an agent's representations. We consider the composition $f : W \to Z, f = h \circ b$. Suppose also that there is a group G of symmetries acting on W via a group action $\cdot_{\mathcal{W}} : G \times W \to W$. A world is thus defined by $(W, \cdot_{\mathcal{W}})$. We would like to find a corresponding group action $\cdot_Z : G \times Z \to Z$ so that the symmetry structure of W is reflected in Z. We also want the group action $\cdot_Z$ to be disentangled, which means that applying G_i to Z leaves all sub-spaces of Z unchanged but the one corresponding to the transformation G_i. Formally, the representation Z is SB-disentangled with respect to the decomposition $G = G_1 \times .. \times G_n$ if:

1. There is a group action $\cdot_Z : G \times Z \to Z$.
2. The map $f : W \to Z$ is equivariant between the group actions on W and Z:

$$\boxed{g \cdot_Z f(w) = f(g \cdot_W w)} \quad \leftrightarrow \quad \boxed{\begin{array}{ccc} G \times W & \xrightarrow{\cdot_W} & W \\ {\scriptstyle id_G \times f} \downarrow & & \downarrow {\scriptstyle f} \\ G \times Z & \xrightarrow[\cdot_Z]{} & Z \end{array}}$$

3. There is a decomposition $Z = Z_1 \times .. \times Z_n$ such that each Z_i is fixed by the action of all $G_j, j \neq i$ and affected only by G_i.

This definition of SB-disentangled representations does not make any assumptions on what form the group action should take when acting on the relevant disentangled vector subspace. However, many subsequent tasks may benefit from a SB-disentangled representation where the group actions transform their corresponding disentangled subspace linearly. Such representations are termed linear SB-disentangled representations, which we refer to as LSB-disentangled representations.

4.3 Contribution: SBDRL requires interaction with environments

4.3.1 Abstract

Finding a generally accepted formal definition of a disentangled representation in the context of an agent behaving in an environment is an important challenge towards the construction of data-efficient autonomous agents. [68] recently proposed Symmetry-Based Disentangled Representation Learning, a definition based on a characterization of symmetries in the environment using group theory. We build on their work and make observations, theoretical and empirical, that lead us to argue that Symmetry-Based Disentangled Representation Learning cannot only be based on static observations: agents should interact with the environment to discover its symmetries. Our experiments can be reproduced in Colab [1] and the code is available[2].

4.3.2 Introduction

We build on the work of [68] and make observations, theoretical and empirical, that lead us to argue that SBDRL requires interaction with environments. The necessity of having interaction has been suggested before [165]. We propose a proof for SBDRL.

As in the original work, we base our experiments on a simple environment, where we can formally define and manipulate a SB-disentangled representation. This simple environment is 2D, composed of one circular agent on a plane that can move left-right and up-down (Figure 4.1). We show that current approaches to learn (L)SB-disentangled representation are not designed to model the effect of the world's symmetries on the representation, a key aspect of SBDRL which we present later. We thus ask: how is one supposed to, in practice, learn a (L)SB-disentangled representation?

We propose two approaches that arise naturally, one where representation and world symmetries effect on it are learned separately and one where they are learned jointly.

[1] `https://colab.research.google.com/drive/1KVlSV24c687N_4TLJWwGTkjt3sh9ufWW`

[2] `https://github.com/Caselles/NeurIPS19-SBDRL`

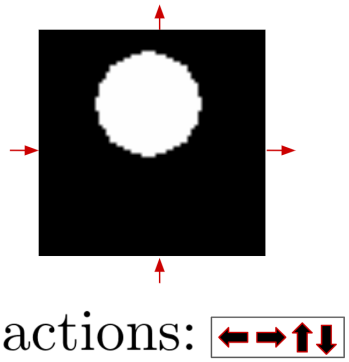


Figure 4.1: Environment studied in this chapter.

For both scenarios, we formally define what could be a proper representation to learn, using the formalism of SBDRL. We propose empirical implementations that are able to successfully approximate these analytically defined representations. Both empirical approaches make use of transitions (o_t, a_t, o_{t+1}) rather than still observations o_t, which validates the main point of this contribution: Symmetry-Based Disentangled Representation Learning requires interaction with the environment.

Ultimately, the goal of such representations is to facilitate the learning of downstream tasks. We study the efficiency of (L)SB-disentangled representation on a particular downstream task: learning an inverse model. Our results suggest that (L)SB-disentangled indeed facilitates the learning of such downstream tasks.

Our contributions are therefore the following:

- We prove that interaction with the environment, i.e. the use of transitions, is necessary for SBDRL, and illustrate it empirically.
- We propose two alternatives for learning linear and non-linear SB-disentangled representation in practice, both using transitions rather than still observations. Using a simple environment, we describe both solutions theoretically and validate them empirically.
- We empirically demonstrate the efficiency of using SB-disentangled representation for a downstream task (learning an inverse model).

4.3.3 Symmetry-Based Disentangled Representation Learning requires interaction with environments

In this section we prove the main claim of this contribution: SBDRL requires interaction with environments. By "interaction with environments" we refer

to the fact that in order to learn a SB-disentangled representation, one should **not** use a training set composed of still samples $(o_t, o_{t+1}, ...)$, but rather transitions $((o_t, a_t, o_{t+1}), (o_{t+1}, a_{t+1}, o_{t+2}), ...)$.

We begin by observing that SBDRL definition is actually two-fold. The definition of a SB-disentangled representation w.r.t the decomposition $G = G_1 \times .. \times G_n$ is composed of two main properties:

1. There is a group action $\cdot_Z : G \times Z \to Z$.
2. The map $f : W \to Z$ is equivariant between the group actions on W and Z.

} Definition of a Symmetry-Based representation.

3. There is a decomposition $Z = Z_1 \times .. \times Z_n$ such that each Z_i is fixed by the action of all $G_j, j \neq i$ and affected only by G_i.

} Disentanglement property.

The first two points define what a SB representation is. It's a representation for which the effect of group actions on the world state is the same as the effect on the representation itself. The third point characterizes what disentanglement is for a SB representation.

In practice, it seems natural to first know how to learn a representation that satisfies the first two points, i.e. a SB representation. Based on this, we can develop methods that enforce disentanglement.

Hence we ask, how can one learn a SB representation? This task involves knowledge about how the group action affects Z. The group action is defined to be the effect of symmetries on the representation. These symmetries can be spatial translations, rotations, time translations, etc. In a Machine Learning paradigm, we would design an algorithm that learns from examples. We thus need, in practice, a way to apply these transformations on observations of the world $(o_t)_{t=1..n}$ and observe the result $(g_t \cdot_Z o_t = o_{t+1})_{t=1..n}$.

We thus make an analogy between the effect of a symmetry g (by the group action $\cdot_\mathcal{W}$) on the environment $(o_1, g, g \cdot_\mathcal{W} o_1 = o_2)$, and a transition that uses the dynamics f of the environment $(o_t, a_t, f(o_t, a_t) = o_{t+1})$. It allows us to consider a more realistic scenario where we have an agent in an environment, and we can apply the group actions to this agent. In our analogy we simply say that $o_1 = o_t$, $o_2 = o_{t+1}$ and $a_t = g$ and $\cdot_\mathcal{W} = f$.

However, we do not make a total confusion between symmetries and regular actions that can be found in any environment. A symmetry is an element of a group (in the mathematical sense) of functions $g : W \to W$, and the binary operation of the group is composition. In that sense, these functions can effectively be considered as actions, because actions take the environment from one state to another through the dynamics f, and symmetries take the environment from one state to another through the group action $\cdot_\mathcal{W}$.

It is important to mention that not all actions are symmetries, for instance the action of eating a collectible item in the environment is not part of any group of symmetries of the environment because it might be irreversible.

More formally, Theorem 1 provides a mathematical proof that we need interaction with environments.

Theorem 1. *Suppose we have a SB representation $(f, \cdot_Z)$ of a world $\mathcal{W}_0 = (W = (w_1, .., w_m) \in \mathbb{R}^{m \times d}, \cdot_{\mathcal{W}_0})$ w.r.t to $G = G_1 \times ... \times G_n$ using a training set $\mathcal{T}$ of unordered observations of $\mathcal{W}_0$. Let W_k be the set of possible values for the k^{th} dimension of $w \in W$.*
Then:

1. *There exists at least $k_{W,G} = n[(\min_k(card(W_k))!] - 1$ worlds $(\mathcal{W}_1, .., \mathcal{W}_{k_{W,G}})$ equipped with the same world states $\mathcal{W}_i = (w_1, .., w_m)$ and symmetries G, but different group actions $\cdot_{\mathcal{W}_i}$.*
2. *For these worlds, $(f, \cdot_Z)$ is not a SB representation.*
3. *These worlds can produce exactly the same training set $\mathcal{T}$ of still images.*

Proof. We prove the three points.

1. For each symmetry G_i, we can shuffle the order of states along each axis of W. For instance, if the symmetry is translation along a cyclic hue axis composed of three colors (red, green, blue). Then one can consider two worlds where translating right from red moves the agent in blue (world 1) or green (world 2).

We provide a lower bound to the number of possible worlds. For a symmetry G_i, the minimal number of possible visited states is $\min_k(card(W_k))$. It is the case if all symmetries affect only one axis of W and all axes of W have the same number of possible values ($= \min_k(card(W_k))$. The number of possible worlds is then given by the number of permutations of a set composed of $\min_k(card(W_k))$ elements, which is $\min_k(card(W_k))!$.

There are n symmetries in $G = G_1 \times .. \times G_n$, hence there are at least $k_{W,G} = n[(\min_k(card(W_k))!] - 1$ possible worlds $(\mathcal{W}_1, .., \mathcal{W}_{k_{W,G}})$ that are not $\mathcal{W}_0$ but share the same state space W and symmetries G. They differ by the action $\cdot_{\mathcal{W}_i}$ of G on the world $\mathcal{W}_i$.

2. For any different world $\mathcal{W}_i$ than $\mathcal{W}_0$, there exists a state and a symmetry $(g, w \in G \times W)$ such that the action of g on w is not the same on the two worlds. Thus, f is not equivariant between the group actions on W and Z w.r.t to both $\mathcal{W}_0$ and $\mathcal{W}_i$. Hence $(f, \cdot_Z)$ is necessarily not a SB representation w.r.t to any of the worlds $(\mathcal{W}_1, .., \mathcal{W}_{k_{W,G}})$ and G.

Formally, let $i \in [|1..k_{W,G}|]$. $\mathcal{W}_i \neq \mathcal{W}_0 \implies \exists (g, w \in G \times W),\ g \cdot_{\mathcal{W}_i} w \neq g \cdot_{\mathcal{W}_0} w$. Necessarily, $f(g \cdot_{\mathcal{W}_i} w) \neq f(g \cdot_{\mathcal{W}_0} w)$. Yet, $(f, \cdot_Z)$ is SB w.r.t $\mathcal{W}_0$: $f(g \cdot_{\mathcal{W}_0} w) = g \cdot_Z f(w)$. Hence, $f(g \cdot_{\mathcal{W}_i} w) \neq g \cdot_Z f(w)$, i.e. for world $\mathcal{W}_i$, $(f, \cdot_Z)$ is not equivariant between the group actions on W and Z.

3. $(\mathcal{W}_0, .., \mathcal{W}_{k_{W,G}})$ all share the same state space. Hence they can theoretically produce any training set of still images collected in $\mathcal{W}_0$.

□

Using Theorem 1, we can deduce that for a given dataset of still images collected in a world, it is impossible to describe the action of symmetries on the world. The dataset could come from a number of different worlds where symmetries act differently. Hence the need for transitions. For example, in a world where the agent can change color along a hue axis, the succession of colors can be (red, green, blue, red, ...), or (red, blue, green, red, ...). Then the world states are identical, the symmetries also. Yet, the effect of the symmetries are not the same, i.e. $\cdot_{1,W} \neq \cdot_{2,W}$.

Still, it is not clear how to discover the symmetries G of a world. [68] propose to use active perception or causal manipulations of the world to empirically determine them. Having this in mind, we note that high-level actions in an environment often correspond to symmetries, such as translations along cartesian axis, rotations, changes of color, changes related to time (no-op action). Actions could then be used as replacements to symmetries, and one could learn SB representations using traditional transitions $(o_t, a_t, o_{t+1})_{t=1..n}$ that are readily available in most environments. In the rest of the chapter, we validate this approach empirically.

4.3.4 Considered environment

We consider a simplification of the environment studied in [68]. This environment is 2D, composed of one circular agent on a plane that can move left-right and up-down, see Fig.4.1.

Whenever the agent steps beyond the boundary of the world, it is placed at the opposite end (e.g. stepping up at the top of the grid places the object at the bottom of the grid). The world-states can be described in two dimensions: (x, y) position of the agent. All of our experimental results are based on this environment. It is simple, yet presents the basis for a navigation environment in 2D. We chose this environment because we are able to theoretically define SB-disentangled representations, without making any approximation. We implement this simple environment using Flatland [23]. The code is available in Colab[3] and Github[4].

4.3.5 Theoretical analysis

We first provide a theoretical analysis of what can be learned in the considered environment, in the formalism of SBDRL. Learning a non-linear

[3] `https://colab.research.google.com/drive/1KVlSV24c687N_4TLJWwGTkjt3sh9ufWW`

[4] `https://github.com/Caselles/NeurIPS19-SBDRL`

SB-disentangled representation of dimension 2 is possible. If (x, y) is the position of the object, then learning these two coordinates as well as the cyclical effect of translations is enough to create a SB-disentangled representation of dimension 2.

However, this is not the case for LSB-disentangled representations. We provide a theorem that proves it is impossible to learn a LSB-disentangled representation of dimension 2 in the environment presented in Sec.4.3.4 (the result also applies to the environment considered in [68]). The key element of the proof is that the two actual dimensions of the environment are not linear but cyclic. Hence the impossibility of modelling two cyclic dimensions using two linear dimensions.

Based on this, we show next how to learn, in practice, a SB-disentangled representation of dimension 2 and a LSB-disentangled representation of dimension 4.

4.3.6 Symmetry-Based Disentangled Representation Learning in practice

We now consider the problem of learning, in practice, SB-disentangled and LSB-disentangled representations for the world considered in Sec.4.3.4. For that, we propose two approaches, either decoupled or end-to-end.

We illustrate each method by learning a SB-disentangled representation with the decoupled approach, and learning a LSB-disentangled representation with the end-to-end approach.

Decoupled approach (illustrated on SB-disentangled representation)

We propose to learn the representation first, and then the group action of G on Z using a separate model. This way, we have a complete description of the SB-disentangled representation. This approach is effectively decoupling the learning of physics from vision as in [60], where there is a vision component that describes and summarizes what the agent sees, a forward module that allows to predict future observations given actions, and a controller module that learns a policy that solves the specified task.

We consider learning a 2-dimensional SB-disentangled representation. We started by reproducing the results in [68]: we used a variant of the current state-of-the-art disentangled representation learning model CCI-VAE [69]. It is a modification of the VAE objective that encourages disentanglement by constraining the shape of the learned latent space to have more efficient representation. The learned representation corresponds (up to a scaling factor) to the world-state W, i.e. the (x, y) position of the agent. This intuitively seems like a reasonable approximation to a disentangled representation.

However, once the representation is learned, we have no idea how the group action of symmetries affect the representation, even though it is at the core of the definition of SBDRL. This is where the necessity for transitions $(o_t, a_t, o_{t+1})_{t=1..n}$ rather than still observations $(o_t)_{t=1..n}$ comes into play. We learn the group action on Z $\cdot_Z : G \times Z \to Z$, such that $f = h \circ b$ is an equivariant map between the actions on W and Z.

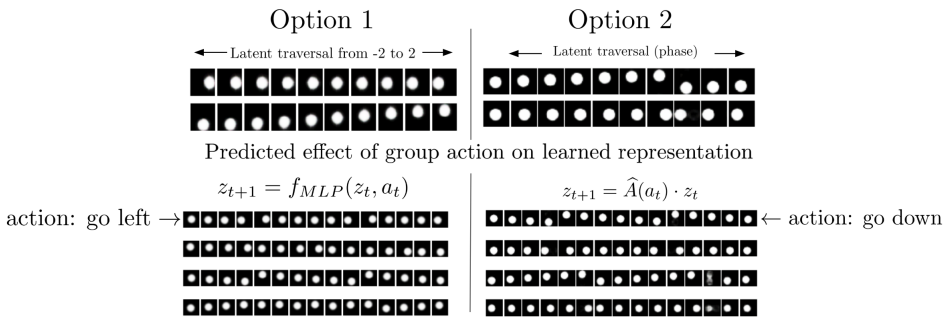


Figure 4.2: **Left:** First option: decoupled learning of representation and group action, here applied to learning a non-linear SB-disentangled representation. Latent traversal spanning from -2 to 2 over each of the representation's dimensions, followed by the predicted effect of the group action associated with each action (left, right, down, up). **Right:** Second option: joint learning of representation and group action, here applied to learning a non-linear LSB-disentangled representation. The representation is complex: latent traversal over the phase of each of the representation's dimensions, followed by the predicted linear effect of the group action associated with each action (down, left, up, right).

In practice, we learn $h : O \to Z$ with a variant of CCI-VAE as explained above, and then use a multi-layer perceptron to learn the group action on Z. The results are presented in Fig.4.2, where we observe that the learned group action correctly approximates the cyclical movement of the agent. We thus have learned a properly SB-disentangled representation of the world, w.r.t to the group decomposition $G = G_x \times G_y$ (decomposition of the transformations possible in the world by the agent as translations along the x and y axis).

End-to-end approach (illustrated on LSB-disentangled representation)

In the decoupled approach, the learned representation is identical to a setting where we would have ignored the group action. Hence, a preferable approach would be to jointly learn the representation and the group action. We study such an approach on the task of learning a LSB-disentangled representation.

To accomplish this, we start with a theoretically constructed LSB-disentangled

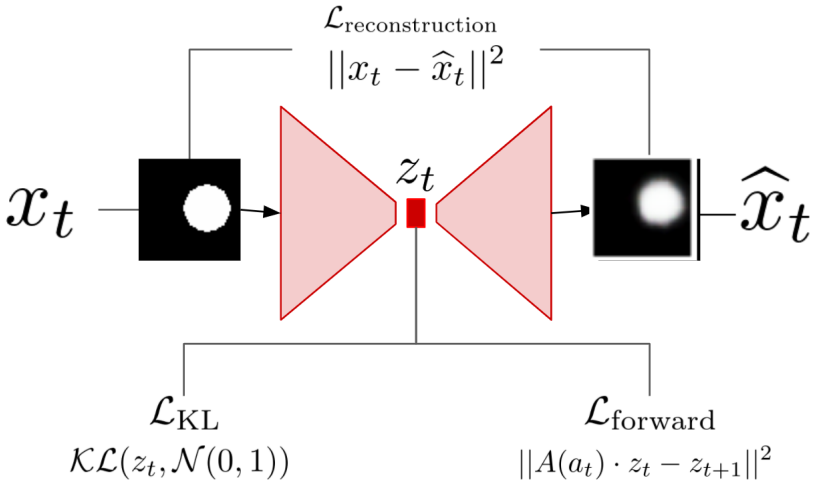


Figure 4.3: Proposed architecture for learning a LSB-disentangled representation in the environment at the left as presented in section 4.3.6.

representation. It is based on an example given in [68]. The representation is defined as following, using 4 dimensions:

- $f : \mathbb{R}^2 \to \mathbb{C}^2$ is defined as $f(x, y) = (e^{2i\pi x/N}, e^{2i\pi y/N})$
- $\rho(g) : \mathbb{C}^2 \to \mathbb{C}^2$ is defined as $\begin{cases} \rho(g_x)(z_x, z_y) = & (e^{2i\pi n_x/N} z_x, z_y) \\ \rho(g_y)(z_x, z_y) = & (z_x, e^{2i\pi n_y/N} z_y) \end{cases}$

In this representation, the (x, y) position is mapped to two complex numbers (z_x, z_y). For each translation (on the x-axis or y-axis), the associated group action on Z is a rotation on a complex plane associated with the specific axis. This representation linearly accounts for the cyclic symmetry present in the environment. It means that the operator that represents the action of the group is linear. Using CCI-VAE with 4 dimensions fails to learn this representation: we verified experimentally that only 2 dimensions were actually used when learning (for encoding the (x, y) position), and the two remaining were ignored.

In order to learn the LSB-disentangled representation, we generate a dataset of transitions, and use it to learn the 4-dimensional LSB-disentangled representation with a specific VAE architecture we term Forward-VAE. This architecture allows to jointly learn the representation and the group action on it. Here, we want the group action on Z to be linear, so we enforce linearity in transitions in the representation space.

We begin by re-writing the complex-valued function $\rho(g) : \mathbb{C}^2 \to \mathbb{C}^2$ as

a real-valued function:

$$\rho(g) : \begin{array}{c} \mathbb{R}^4 \to \mathbb{R}^4 \\ v \to \rho(g)(v) = A^*(g) \cdot v \end{array} \tag{4.1}$$

where $A^*(g)$ is a 4x4 block-diagonal matrix, composed of 2x2 rotation matrices. Let's consider the environment in Sec.4.3.4. The agent has 4 actions: go left, right, up or down. We associate each action with a corresponding matrix with trainable weights.

For instance, if $g = g_x \in G_x$ is a translation on the x-axis, the corresponding matrix is $A^*(g_x)$ and we associate actions go right/left with corresponding matrices $\hat{A}(a_t)$, where $\cdot$ are trainable parameters:

$$A^*(g_x) = \begin{bmatrix} \cos(n_x) & -\sin(n_x) & 0 & 0 \\ \sin(n_x) & \cos(n_x) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \text{ and } \hat{A}(g_x) = \begin{bmatrix} \cdot & \cdot & 0 & 0 \\ \cdot & \cdot & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

We would like the representation model that we learn to satisfy $\rho(g)(v_t) = \hat{A}(g) \cdot v_t = v_{t+1}$. We thus enforce the representation to satisfy it in our Forward-VAE architecture, as illustrated in Fig.4.3. The training procedure is presented in Algorithm 1 thereafter.

For each image in a batch, we compute $f(o_t) = z_t$ and $f(o_{t+1}) = z_{t+1}$ using the encoder part of the VAE. Then we decode z_t with the decoder and compute the reconstruction loss $\mathcal{L}_{reconstruction}$ and annealed KL divergence $\mathcal{L}_{KL}$ as in [25]. Then we compute $\hat{A}(a_t) \cdot z_t$ and compute the forward loss, which is the MSE with z_{t+1}: $\mathcal{L}_{forward} = (\hat{A}(a_t) \cdot z_t - z_{t+1})^2$. We then backpropagate w.r.t to the full loss function of Forward-VAE:

$$\mathcal{L}_{Forward-VAE} = \mathcal{L}_{reconstruction} + \gamma_t \cdot \mathcal{L}_{KL} + \mathcal{L}_{forward} \tag{4.2}$$

The results are presented in Fig.4.2. Forward-VAE correctly learns a representation where the two complex dimensions correspond to the position (x, y) of the agent. Plus, we observe that the learned matrices $(\hat{A}_i)_{i=1..4}$ are very good approximation of the ideal matrices $(A^*_i)_{i=1..4}$ defined above, with $n_x \approx \frac{\pi}{3}$. The mean squared difference is very small (order of 10^{-4}).

Remarks

Note that we could have applied this joint learning approach to learning non-linear SB-disentangled representation. However it is not possible to apply the decoupled approach to learning a LSB-disentangled representation.

We used inductive bias given by the theoretical construction of a LSB-disentangled representation theory to design the action matrices and its trainable weights. This construction is specific to this example. However, the idea of having an action matrix for each action is extendable. If each action is high-level and associated with a symmetry, then SBDRL can be

Algorithm 1 Pseudo-code for training procedure of Forward-VAE

batch $= ((o_t, .., o_{t+k}), (a_t, .., a_{t+k-1})) = (\mathbf{o}, \mathbf{a})$
for batch in dataset **do**

— Forward model Loss—

$\mathbf{z} \leftarrow encoder\_mean(batch)$
$\mathbf{z_{before}} \leftarrow z[: -1]$
$\mathbf{z_{after}} \leftarrow z[1 :]$ *# targets*
$\hat{\mathbf{A}} \leftarrow [\hat{A}(a_t), .., \hat{A}(a_{t+k-1})]$ *# actions matrices corresponding to given action sequence*
$\mathbf{z_{prediction}} \leftarrow \hat{\mathbf{A}} \cdot \mathbf{z_{before}}$ *# predictions*
$\mathcal{L}_{forward}(batch) \leftarrow MeanSquaredError(\mathbf{z_{prediction}}, \mathbf{z_{after}})$

— VAE Loss (reconstruction and KL) —

$\mathbf{z} \leftarrow encoder\_sample(batch)$
$\hat{\mathbf{o}} \leftarrow decoder(\mathbf{z})$
$\mathcal{L}_{recon}(batch) \leftarrow MeanSquaredError(\hat{\mathbf{o}}, \mathbf{o})$
$\mathcal{L}_{KL}(batch) \leftarrow KL\_divergence(\mathbf{z}, \mathcal{N}(0, 1))$

— Backpropagation —

$\mathcal{L}_{Forward-VAE}(batch) \leftarrow \mathcal{L}_{recon}(batch) + \mathcal{L}_{KL}(batch) + \mathcal{L}_{forward}(batch)$
$encoder, decoder, (\hat{A}_1, .., \hat{A}_j) \leftarrow Backpropagation(\mathcal{L}_{Forward-VAE}(batch))$

performed. Still, it requires high level actions that represent these symmetries. One potential way to find these actions is through active search [159], as suggested in [68].

In our Forward-VAE architecture we indeed explicitly design the model such that the resulting representation is Linear SB-disentangled, because we enforce linearity, force the representation to be SB (see points 1 and 2 in the definition in Sec.3) and by design have two separate subspaces for each symmetry. A more general approach would have been not to have those two separated subspaces and learn the entire action matrices, and thus we won't have the guarantee that the representation will satisfy the disentangled property. We ran this experiment and obtained the expected result: the learned representation is Linear-SB but not disentangled. This means that the x and y coordinates are not properly disentangled w.r.t to the considered group decomposition (i.e. a latent traversal over each dimension would not result in only a movement of the agent along the x or y coordinate). However, the learned action matrices are able to describe how the symmetries affect the representation in a linear way. Hence enforcing disentanglement is the only viable option we found for LSB-disentanglement with this architecture.

It is important to note that an instability in Forward-VAE training can be expected due to the different contributions of the loss: at each training steps the goal of the forward part of the loss is to have a latent space that is suited for predicting z_{t+1} using z_t. The rest of the loss is the VAE, which tries to learn a latent space that allows reconstruction. Hence the balance between these two seemingly unrelated objectives might be a source of instability. However it worked in practice, without any re-weighting of the objectives, which was a surprise.

4.3.7 Using (L)SB-disentangled representations for downstream tasks

Is using (L)SB-disentangled representations beneficial for subsequent tasks? This remains to be demonstrated, as other work has already challenged the benefit of learning disentangled representations over non-disentangled ones [105]. In this section we wish to answer the following question: **is it increasingly better to use non-disentangled / non-linear SB-disentangled / LSB-disentangled representation for downstream tasks?** We define better in terms of final performance, under different settings (restricted capacity classifiers/restricted amount of data).

For the choice of downstream tasks, we select the task of learning an inverse model, which consists in predicting the action a_t from two consecutive states (s_t, s_{t+1}).

As a LSB-disentangled representation models the interaction with the environment linearly, it intuitively should be increasingly easier to learn an

inverse model from: a non-disentangled representation, a non-linear SB-disentangled representation, and a LSB-disentangled representation.

Experimental protocol

In order to test this hypothesis, we selected a well-established implementation (Scikit-learn [129]) of a well-studied classifier (Random Forest [15]). We collect 10k transitions (o_t, a_t, o_{t+1}). We train the following models and baselines to compare:

- LSB-disentangled representation of dimension 4: Forward-VAE trained as in Sec.4.3.6.
- SB-disentangled representation of dimension 2: CCI-VAE variant trained as in Sec.4.3.6.
- Non-disentangled representation of dimension 2: Auto-encoder, non-disentangled baseline.
- SB-disentangled representation of dimension 4: CCI-VAE trained as in Sec.4.3.6 but with 4 dimensions, baseline to control for the effect of the size of the representation.

For each model, once trained, we created a dataset of transitions in the corresponding representation space (s_t, a_t, s_{t+1}). We then report (Figure 4.4) the 10-fold cross-validation mean accuracy as a function of the maximum depth parameter of random forest, which controls the capacity of the classifier.

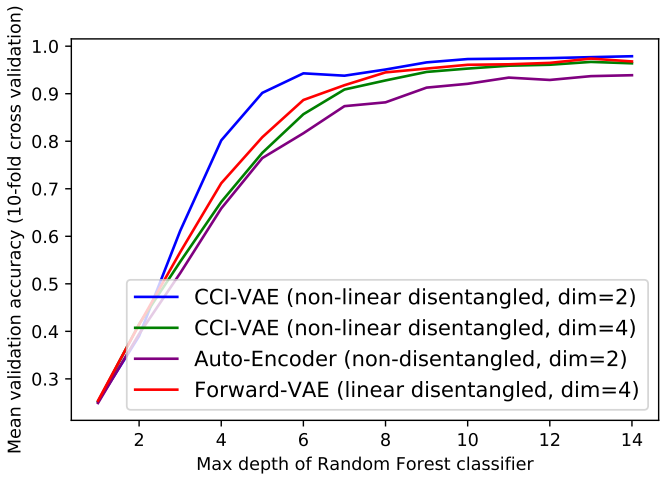
Results

We first observe that in all cases, either LSB or SB-disentangled representations are performing best. In terms of final performance, all models meet at the upper 100% accuracy limit, given enough data and a classifier with enough capacity.

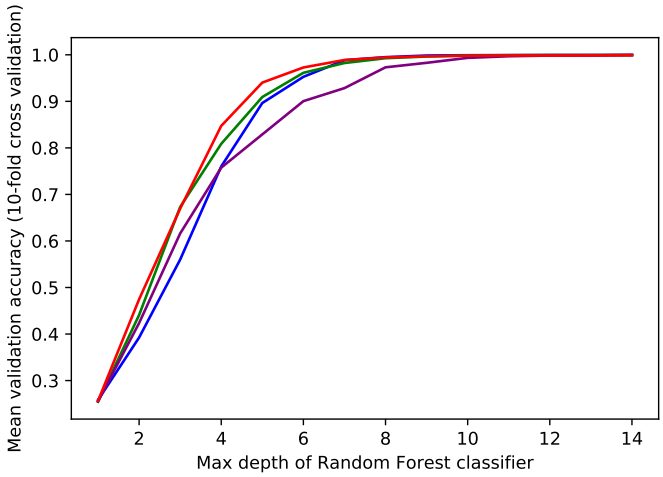
However, if we consider a constraint in training set size and a fixed high capacity classifier (see Fig. 4.4(a)), we can see that using a SB-disentangled representation is superior to other options. We refer to the capacity of the classifier as "high" if increasing the capacity parameter does not lead to an increase in validation accuracy.

Moreover, if we consider a fixed large training set size and a constraint on the classifier's capacity, using LSB-disentangled representation is the best option (see Fig. 4.4(b)).

As a conclusion, we observed that it is easier for a small capacity classifier to solve the task using a LSB-disentangled representation and it is easier to solve the task using less data with a SB-disentangled representation. This



(a) Dataset size: 1k samples



(b) Dataset size: 10k samples

Figure 4.4: Downstream task evaluation of representation models: inverse model prediction. Mean 10-fold cross validation accuracy as functions of dataset size and classifier capacity (max depth parameter of Random Forest). LSB and SB-disentangled representation perform best.

indicates that (L)SB-disentanglement is indeed useful for downstream task solving.

Remarks

It's worth noting that the advantage is not very substantial, which is expected due to the simplicity of the task. Our results on usefulness of (L)SB-

disentangled representations for downstream tasks are preliminary; it would be interesting as future work to compare to more baselines and on more tasks. Other related works such as [169, 103] also study the usefulness of disentangled representations for downstream tasks, and respectively find them useful for performance in abstract visual reasoning tasks and for encouraging fairness when sensitive variables are not observed.

More generally, we need large-scale evaluations of representations' usefulness for downstream tasks in the disentanglement representation learning literature, like the study proposed in [169]. Such studies are needed to validate the intuition that disentanglement is useful in practice for subsequent tasks.

4.3.8 Discussion

The benefit of using transitions rather than still observations for representation learning in the context of an agent acting in an environment has been proposed, discussed and implemented in previous work [165, 136]. In this work however, we emphasize that using transitions is not only a beneficial option, but is compulsory in the context of the current definition of SBDRL for an agent acting in an environment, as Theorem 1 proves it.

Applying SBDRL to more complex environments is not straightforward. For instance, consider that we add an object in the environment studied in this chapter. Then the group structure of the symmetries of the world are broken when the agent is close to the object. However, the symmetries are conserved locally. One approach could be to start from this local property to learn an approximate SB-disentangled representation.

4.4 Conclusion

In this chapter, we studied and proved the necessity of actions for disentanglement in Symmetry-Based Disentangled Representation Learning. More generally, the need for actions is advocated in most theories of perception. Since disentanglement is a supposed feature of our perception, it is logical to find the need for actions in disentanglement learning. We were able to prove it for SBDRL, and provide empirical evidence of the usefulness of such disentangled representation for downstream tasks.

In the next chapter, we will continue by taking inspirations from two theories of visual perception that put an emphasis on the crucial role of actions in the learning of perception. From these two theories we will introduce the concept of sensory commutativity of action sequences, which describes the commutative properties of action sequences with respect to sensory information received by the agent. We will first study it theoretically, and then provide concrete application algorithms.

Chapter 5

Sensory commutativity of action sequences: theory

Contents

5.1 Introduction

In the rest of this thesis, we present approaches that are inspired from the theories of embodied perception, and implemented using or thanks to recent ML advances. This approach is a natural progress in the quest for solving the problem of perception: new tools and learning mechanisms are available

in the literature, and they can thus be used to implement intuitions or concepts that have been developed concurrently in philosophy, psychology or cognitive sciences.

We now present our contribution to this effort. We take inspiration from the Sensorimotor contingencies theory (SMCT) theory [123] and Gibson's visual perception theory [134, 132, 53] to develop theoretical insights on perception and learning algorithms based on the concept of commutativity of action sequences with respect to sensory information received by the agents while they act in the environment. Chapter 5 presents our theoretical findings while Chapter 6 introduces our proposed learning algorithms based on the theory developed.

5.2 Sensory commutativity of action sequences: motivation

Sensorimotor contingencies theory (SMCT) [123] is a theory of visual perception developed by J. Kevin O'Regan that gives a center role to actions in the development of visual awareness. Many current neurophysiological, psychophysical, and psychological approaches to vision rest on the idea that when we see, the brain produces an internal representation of the world. The activation of this internal representation is assumed to give rise to the experience of seeing. O'Reagan proposes that seeing is a way of acting. He claims that activity in internal representations does not generate the experience of seeing. In his theory, the outside world serves as its own, external, representation. The experience of seeing occurs when the organism masters what he calls the governing laws of sensorimotor contingencies. This approach has the advantage to provide a natural and principled way of accounting for visual consciousness, and for the differences in the perceived quality of sensory experience in the different sensory modalities. Several lines of empirical evidence have brought support to the theory, in particular evidence from experiments in sensorimotor adaptation, visual "filling in," visual stability despite eye movements, change blindness, sensory substitution, color perception, and space awareness [91, 93, 92, 90, 132, 131, 133, 28, 27]. Moreover, in this theory, sensory compensability of actions is a key to the understanding of the concept of space.

The idea of compensability of actions was already put forward by Poincaré [134], who suggested that the set of compensable transformations of the environment together with the composition operation forms a group. Philipona et al. [131] further attempted at describing this group, by noticing that some transformations can commute while others don't. Using action sequences and their commutative property, the authors suggested that spatial transformations and non-spatial transformations can be disentangled. Moreover, these intuitions and remarks are linked to the study of symmetry-based

disentanglement, that works by separating transformations in groups as well.

We observed that these three perspectives on perception (Poincaré, Philipona, SBDRL) all point to a mathematical formalism using groups, compensability/commutativity, and action sequences. While Poincare and Philipona proposed intuitions without actual implementations, SBDRL demonstrated that the intuition was valid on toy examples. We build on these intuitions to study whether they can transfer to practical applications in the domain of representation learning.

In this chapter, we formalize a measure for sensory commutativity. We then notice that we can deduce practical applications from the study of the commutatitive properties of action sequences. This work is novel since these comments we take inspiration from were not formalized as research work, but rather cited as intuitions that could be interesting.

We thus consider the set of action sequences, termed $Seq(\mathcal{M})$, and their commutative properties. We will now study the group and sub-group properties of $Seq(\mathcal{M})$.

5.3 Commutative properties of action sequences

5.3.1 Formalism choice

In the SMCT theory, the agent sensory motor experience is described as follows:

$$s_t = \phi(m_t, \epsilon_t) \tag{5.1}$$

This formalism, while close to the RL formalism, is centered around the agent and its perception. At a time t, the agent is in a particular motor state m_t. This means that its motors are in a particular setup called m_t (e.g. the actuator's torque and angle). The environment is defined by everything that's not the agent. It's thus an entity that is in a state ϵ_t, e.g. a room with 6 walls plus light sources and objects placed in different locations. The agent can perceive the world through its sensorimotor dependencies ϕ: a function that takes as input m_t and ϵ_t and produces sensory inputs from its sensors s_t.

Next, we would like to describe the dynamics of the world. This description is generally not present in SMT theory. Thus Eq.5.1 is not sufficient to support the description of the dynamics of the world. We propose to model these dynamics with the following equation:

$$m_{t+1}, \epsilon_{t+1} = f(m_t, \epsilon_t, \Delta_{m_t}^{m'_{t+1}}, \Delta_{\epsilon_t}^{\epsilon'_{t+1}}) \tag{5.2}$$

The agent can operate motor commands $\Delta_{m_t}^{m'_{t+1}}$, which will in turn change it's sensory inputs to s_{t+1} through the function ϕ. The environment can

change also and influence the agent, represented by $\Delta_{\epsilon_t}^{\epsilon'_{t+1}}$. Taking the initial states and changes as inputs, the function f yields the new motor command m'_{t+1}, and a new configuration of the environment ϵ'_{t+1}. We don't generally have that $\epsilon_{t+1} = \epsilon'_{t+1}$ or $m_{t+1} = m'_{t+1}$ since the agent can affect the environment configuration through its motor commands or the environment can force movements on the agent.

In summary, by combining Eq.5.1 and Eq.5.2, we obtain an equation that includes the dynamics of the world in classical SMT formulation:

$$s_{t+1} = \phi(m_{t+1}, \epsilon_{t+1}) = \phi(f(m_t, \epsilon_t, \Delta_{m_t}^{m'_{t+1}}, \Delta_{\epsilon_t}^{\epsilon'_{t+1}}))$$

5.3.2 Group structure of the set of action sequences $Seq(\mathcal{M})$

We will now formalize groups and sub-groups of symmetries in the case of an agent moving in its environment. We study the set of motor command (or action) sequences of finite length, referred to as $Seq(\mathcal{M})$, and will attempt to describe its structure.

Philipona [131] first defined a relation between action sequences: $h \sim g$ if and only if h and g affect the sensors in the same way. Using our formalism, we can translate this concept into an equality.

Definition 1. *Let $(h, g) \in Seq(\mathcal{M})$. h is equivalent to g under (m_t, ϵ_t), noted $h \sim_{m_t,\epsilon_t} g$ if and only if they produce the same sensory states when applied from the same starting situation of the agent (m_t) and the environment (ϵ_t):*

$$h \sim_{m_t,\epsilon_t} g \iff \phi(f(m_t, \epsilon_t, h, \Delta_{\epsilon_t}^{\epsilon_{t+1}})) = \phi(f(m_t, \epsilon_t, g, \Delta_{\epsilon_t}^{\epsilon'_{t+1}}))$$

Intuitively, two action sequences are equivalent for a particular motor state and environment state if applying them leads to the same sensory state. For instance in the case of a multiple-joint arm moving freely in an empty space, there are multiple different ways of moving the arm from one motor state to another. This yields action sequences $(h_1, .., h_n)$ which are equivalent in this situation (m_t, ϵ_t), we thus have $h \sim_{m_t,\epsilon_t} g$. However in other situations these action sequences can become not equivalent, for instance if there are objects on the way as illustrated in Fig. 5.1.

For convenience and clarity, we will drop the notation for dependence on (m_t, ϵ_t) and thus write $h \sim g$ whenever there are no ambiguities in the context. We now consider the structure of $Seq(\mathcal{M})$ under composition $\circ$ with respect to the equivalence $\sim$.

Proposition 1 (Structure of $(Seq(\mathcal{M}), \sim, \circ)$)**.** *The following properties hold:*
1. $\sim$ is an equivalence, i.e. it is reflexive, transitive and symmetric.

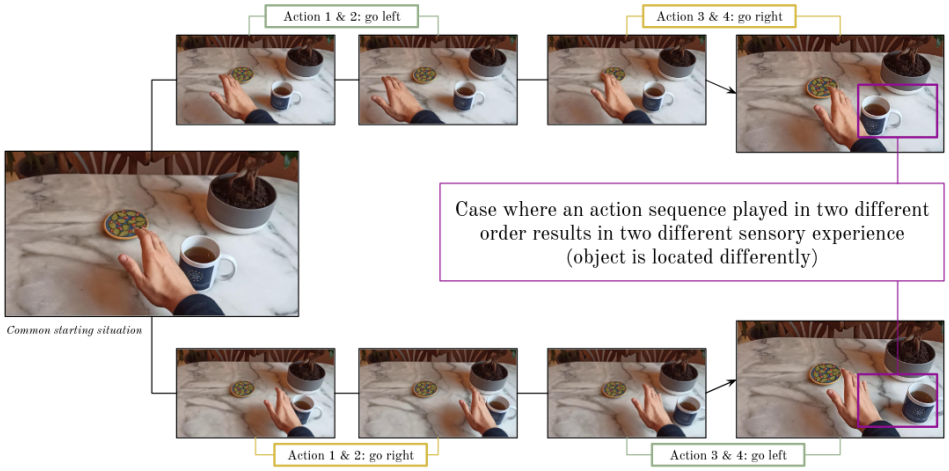


Figure 5.1: Example of action sequences that do not commute. Starting from a common situation, the action sequence played in two different orders does not lead to the same sensory state.

2. $(Seq(\mathcal{M}), \circ)$ *is a group w.r.t* $\sim$.
3. $\circ$ *is not commutative with respect to* $\sim$.

Proof. 1. $=$ is an equivalence, thus $\sim$ is an equivalence as well.

2. All 4 properties of the group definition are satisfied. (i) For two action sequences $(h, g) \in Seq(\mathcal{M})$, the composition of h and g is still an action sequence $h \circ g \in Seq(\mathcal{M})$. (ii) $\circ$ is associative with respect to $=$, i.e. $g \circ (h \circ k) = (g \circ h) \circ k$ thus it follows that $g \circ (h \circ k) \sim (g \circ h) \circ k$. (iii) The identity element is the no-op action. (iv) If we suppose that there are no irreversible phenomenons in the environment, then for a fixed (m_t, ϵ_t), all action sequences can be inverted.

3. $\circ$ is not commutative, as we can always explicitly find two action sequences that do not commute. For instance, once there exists a movable object in the environment: if the agent is placed left to the object, then let h be moving right and g be moving left. h and g do not commute (Fig. 5.1).

□

$(Seq(\mathcal{M}), \circ)$ is thus a group w.r.t $\sim$. This structure is consistent with the intuitions in SBRL and SMT theories. In the following, we build on the observation that composing action sequences is not generally commutative and we can measure to which degree they commute. We will show how this property can lead the agent to organize and interpret its motor space and discover objects in the environment.

5.3.3 Philipona's conjecture

Philipona [131] already studied how action sequences commute with respect to the sensory information received by the agent. Notably, Philipona defined commutative residues. Suppose that an agent doing $h_1 \circ h_2$ leads to a different outcome in observations than doing $h_2 \circ h_1$, then a commutative residue g is an action sequence that the agent has to do to compensate the difference in sensory experience.

Definition 2. *g is a commutative residue of (h_1, h_2) if and only if $h_1 \circ h_2 \sim_s h_2 \circ h_1 \circ g$. If g is equivalent to no-op (no action), then h_1 and h_2 commute.*

Starting from this definition, he conjectured that all action sequences that are not displacements commute with any action sequences. For instance, moving your arms (displacement action) then opening the eyes (non-displacement action) will always commute whereas two displacement actions will not necessarily commute, depending on which starting situation s is selected.

Conjecture 1 (Philipona's conjecture)**.** *Let $Seq(\mathcal{M})$ be the set of action sequences. The subset of $Seq(\mathcal{M})$ composed of non-displacements action sequences is the sub-group of $Seq(\mathcal{M})$ that commutes.*

We will illustrate this conjecture with experiments in Sec. 5.5.3.

5.3.4 SC-experiment definition

Based on Philipona's conjecture, we derive a criterion for characterizing how much each degree of freedom of the agent affects the world, computable using only sensorimotor data. We define "degree of freedom" (DOF) as a dimension of the multidimensional continuous action space of the agent. We also define what we term a sensory commutativity experiment: for an action sequence h, the agent plays it in two different orders starting from the same situation.

Definition 3 (Sensory commutativity experiment (SC-experiment))**.** *Let h be an action sequence of finite length. Let h_p be a random permutation of h (same sequence but different order).*

We define a sensory commutativity experiment (SC-experiment) as playing h and h_p from the same starting point and comparing the two resulting observations in the agent's sensors.

5.4 Sensory commutativity probability of an action sequence

5.4.1 SCP definition

Using Philipona's conjecture, we have that for an SC-experiment, the agent can experience two different sensory outcomes only if the action sequence h is composed of at least one displacement action (an action that affects the environment such as moving limbs or going forward).

However, not all displacement actions are equivalent. The agent is more likely to observe two different outcomes if the action sequence is composed of displacement actions that affect the environment *a lot*. Consider moving your forearm (elbow joint) compared to moving your whole arm (shoulder joint): the latter is more likely to move things around in the environment and thus induce sensory non-commutativity when played in two different orders (i.e. having two different sensory outcomes). Actions on an elbow joint should therefore have a higher probability to lead to non-commutativity than a shoulder joint.

We formalize this intuition by defining the Sensory Commutativity Probability (SCP) of a degree of freedom, averaged over all starting situations s:

Definition 4 (Sensory commutativity probability of a degree of freedom)**.** *Let $Seq(\mathcal{M}_k)$ be the set of motor commands (or action) sequences of finite length for the k^{th} degree of freedom of $\mathcal{M}$ (motor state space). Let $h \in Seq(\mathcal{M}_k)$ and let h_p be a random permutation of h (same sequence but different order).*

The Sensory Commutativity Probability of the k^{th} degree of freedom $SCP(\mathcal{M}_k)$ is defined as:

$$SCP(\mathcal{M}_k) = \mathbb{P}_{s,h}[h \sim_s h_p]$$

5.4.2 SCP computation

We propose a straightforward procedure to estimate the SCP of each degree of freedom of the agent. We initialize the SCP value to 0 (`SCP←0`). We then repeat the following process n times for each DOF:

- Sample an action sequence using the selected degree of freedom (a sequence of action where each action is a value between -1 and 1).

- Play it in 2 different orders starting from the same randomly chosen state and save the two final sensor images s_1 and s_2. Compute the distance between the two images $d(s_1, s_2)$.

- Count one (`SCP+=1`) if $d(s_1, s_2) \leq \tau$, zero otherwise.

Finally, the estimator of the SCP is the average over the number of trials (`SCP←SCP/`n).

The parameters of the algorithm are the selected distance function d that allows comparing the agent's observations, the threshold τ, and the number of iterations n. Note that using a simulation allows playing the two action sequences of different orders from the exact same starting position.

5.5 SCP experimental analysis

In this experimental section, we compute and interpret the SCP in 2D and 3D embodied agent scenarios. In order to study the properties of SCP and how it relates to the emergence of the notion of objects, we use simulation environments that have the following properties: embodied agent, navigable space with objects to interact with, first-person high-dimensional observations, low-level high-dimensional action space, and coherent physics, as described in Chapter 2.

5.5.1 2D experimental setup

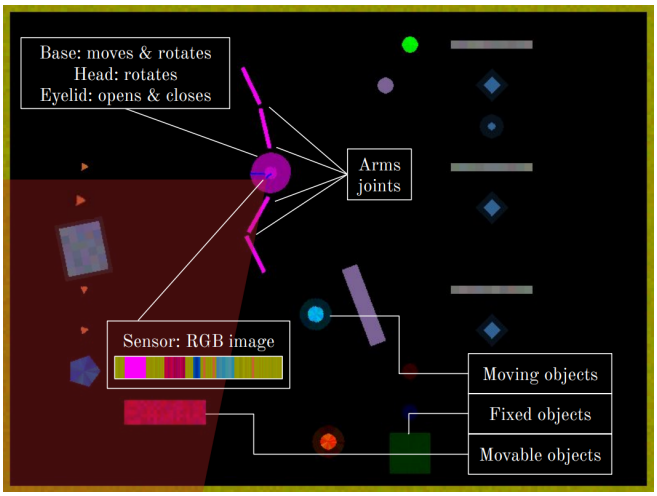


Figure 5.2: Simulation used for our experiments. The agent Polyphemus has a 8 DOF motor space, receives an image of it's only eye, and is placed in a room with fixed, movable and moving elements.

Simulation description. Our first experiment uses Flatland [23]. We construct an agent called Polyphemus (a Cyclop from the Greek mythology), that has a base that can move forward and rotate, a rotatable head and two 2-DOF arms. The agent sees through its unique eye that has an activable eyelid, for a total of 8 DOF. The observation received by the agent is a 64x3 line of RGB pixels (as the world is 2D), which corresponds to the field of view of 90 degrees. This agent is placed in a room with fixed, moving, or movable entities, all of different colors. It can move around and

physically interact with these entities. Its point of view can change through base movement, rotation, and head rotation. Our simulation is illustrated in Fig. 5.2. For each degree of freedom, an action or motor command corresponds to a change in the longitudinal/angular velocity of the degree of freedom.

SCP computation. In order to compute the SCP of each of the 8 agent's degrees of freedom, we have to select a distance and threshold as mentioned in Sec. 5.4.2. The distance selected here is simply the mean squared error between s_1 and s_2, the observations resulting from the two sequences of actions of a SC experiment. Because there is no noise in the dynamics of the environment and the sensor, the future of the agent is deterministic. Therefore, in this particular case we can use a threshold of 0. This means that we consider that two action sequences sensory commutes if and only if applying the two action sequences from the same initial state lead to exactly the same sensors. This hard constraint will be relaxed in subsequent experiments (Sec. 5.5.2).

Baselines. The SCP criterion derived in this thesis estimates how much each degree of freedom affects the environment in an embodied agent scenario. We tried two alternatives to this approach in order to estimate the same quantity. A straightforward approach to this problem, which we call the naive alternative, is to play action sequences of each degree of freedom and quantify how much the sensors change. A more involved approach is to use prediction on the sensory change caused by each degree of freedom, a common approach used to improve exploration in RL [17, 128]. We call this alternative the prediction error approach. The DOF that are harder to predict could be the ones affecting the environment the most, and thus the most important for manipulation and navigation.

5.5.2 3D realistic experimental setup

We also compute and interpret the SCP for a realistic embodied agent scenario using the interactive Gibson environment (iGibson) [177].

Figure 5.3: **Left:** External view of the iGibson simulator where the Fetch robot is in a living room. **Right:** Fetch's first person view.

Simulation description. In our experiments, we use the Rs environment from iGibson, which is basically a regular apartment. We place the

Fetch robot in this environment (Fig. 5.3, left). Fetch is a 10-DOF real robot [174] equipped with a 7-DOF articulated arm, a base with two wheels, and a liftable torso. Fetch perceives the environment through a camera placed in its head (Fig. 5.3, right).

SCP computation. In the Flatland environment, two action sequences commuted only if the sensory result of applying both from the same starting situation were perfectly equal. We relax the strict equality condition to compute the SCP for Fetch. Indeed, with real images, only an offset of one pixel would render the two action sequences non-sensory commutative. Instead of using the mean squared error as a distance, we use a perceptual distance using the VGG16 [157] features of each observation. We thus have $d(s_1, s_2) = ||VGG16(s_1) - VGG16(s_2)||_2^2$. The choice of the threshold τ is partly arbitrary, as we are interested in relative comparisons between degrees of freedom. We verify in our experiments that our results and conclusions are valid for a large range of τ.

5.5.3 Results

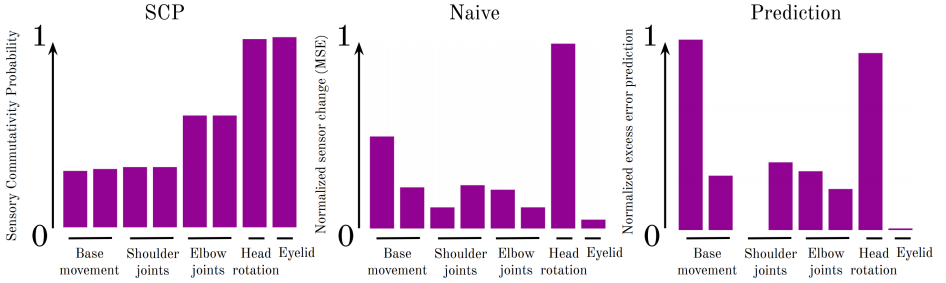


Figure 5.4: Results for the experiments in Flatland. **Left:** Sensory Commutativity Probability for each degree of freedom. **Middle:** Naive alternative. **Right:** Prediction error alternative.

In the Flatland environment, Fig. 5.4 (Left) shows that only two actions have an SCP of 1: *eyelid* and *head rotation*. All other actions have an SCP inferior to 1. **This is consistent with Philipona's conjecture** (Sec. 5.3.3): *eyelid* and *head rotation* are the two degrees of freedom that are not associated with displacements, thus action sequences composed of actions of these type commute with respect to the sensors. On the contrary, all other degrees of freedom are associated with displacements, and thus will eventually induce non-zero commutation residues when played in different orders from the same starting situation. We observe the same results in iGibson, presented in Fig. 5.5: the torso lift DOF is not associated with displacement in the environment, so it has an SCP of 1, i.e. it always sensory commutes. Hence the results are consistent with the conjecture and

can be used by the agent to autonomously discover which of its actions are associated with displacements or not.

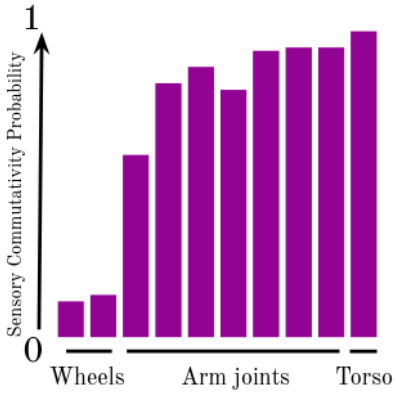


Figure 5.5: SCP computed for each of Fetch's degrees of freedom.

Qualitatively, SCP is inversely proportional to how each degree of freedom affects the environment. By that we mean that from the computation of the SCP, we obtain a hierarchical organization of the action space in which the more important dimensions for manipulation and navigation are separated from the dimensions that are not crucial for such tasks. For instance, we hypothesized that shoulders should have a lower SCP than elbows since activating the shoulder joint is more likely to induce non-commutativity by moving things around or hitting walls/obstacles. This intuition is verified by our results. Shoulders and base movement have a lower SCP than elbows which in turn have a lower SCP than eyelid and head rotation, as observed in Fig. 5.4. Without having any prior knowledge about the simulation, we can automatically organize the agent's degrees of freedom in a hierarchy. Moreover, the symmetry of the action space is kept, as elbow 1 and 2 have equal SCP, and so do shoulder 1 and 2. We reach the same conclusions on iGibson (see Fig. 5.5). The wheels have the lowest SCP since they provide longitudinal movement and rotations for the robot. Then comes the first DOF of the articulated arm, i.e. the ones that are closer to its base (like shoulders vs. elbows in the Flatland experiments). Finally, the highest SCP values correspond to the arm DOF that is further on its arm and the torso lift. Once again, we obtain a hierarchical organization of the action space in which the less important dimensions for manipulation and navigation are separated from the dimensions that are not crucial for such tasks.

About the choice of the threshold to compute the SCP, we tried a range of values for τ, from 20 to 100, and in each case, we obtain the same hierarchy and thus the same conclusion, only the nominal values change, which is

irrelevant for the use of SCP.

We verified the robustness of these results. We computed the SCP for 8 different combinations of agents and environments (longer/smaller arms, more/fewer objects) and confirmed our intuitions on the interpretation of SCP described above. We also verified the robustness of these results in iGibson by computing the SCP for a different type of robot called JackRabbot [110]. We reach the same conclusions as with the Fetch robot.

Alternative methods are not adapted. Results are illustrated in Fig. 5.4. Both approaches fail to replace the SCP criterion. We see that for the naive approach, rotating the head of the agent changes dramatically what the agent sees, even though this degree of freedom does not affect the environment. For the prediction error alternative, we see the same problem with head rotation and a great difference between the two base movements (rotation and longitudinal movement) while they affect the environment in similar ways. Indeed, it's harder to predict what's outside the field of view of the agent so rotation is harder to predict compared to longitudinal movement. To conclude, the proposed alternatives could not yield the same organization of the agent's DOF.

5.6 Conclusion

In this chapter, we introduced Sensory Commutativity of action sequences, a concept derived from the SMCT theory that studies the commutative properties of action sequences with respect to sensory information received by the agent.

We formalized the notion of sensory commutativity, and characterized the properties that emerge from that. The set of action sequences equipped with the equivalence operation and the composition operation is a non-commutative group.

We then translated Philipona's conjecture in our formalism: it states that the subset of action sequences that are not displacements is the subgroup of the set of action sequences that commutes.

We introduced the SC-experiment, which allows testing of sensory commutativity in simulators. It consists in playing an action sequence in two different orders from the same starting point, and then comparing the outcomes in observations.

We finally introduced SCP, a criterion that allows to hierarchically sort the DOF of an agent by relative importance regarding displacements. By using Flatland and iGibson, respectively a 2D and 3D simulation that respects the required features for the study of perception as presented in Chapter 2, we showed that SCP indeed allows to empirically validate the properties described in this Chapter and characterize the motor space of the embodied agent, based only on naive exploration of the environment.

We will now present our implementations of the SCP, its interest for speeding up RL, as well as an object detection method based on SC-experiment called SCOD.

Chapter 6

Sensory commutativity of action sequences: applications

Contents

In order to illustrate all the concepts introduced in the previous chapter, the experiments presented now are organized as follows: we show how we can use SC-experiments to learn about immovable and movable objects in realistic robotics setups. Finally, we show how SCP can be used for improving sample-efficiency in RL.

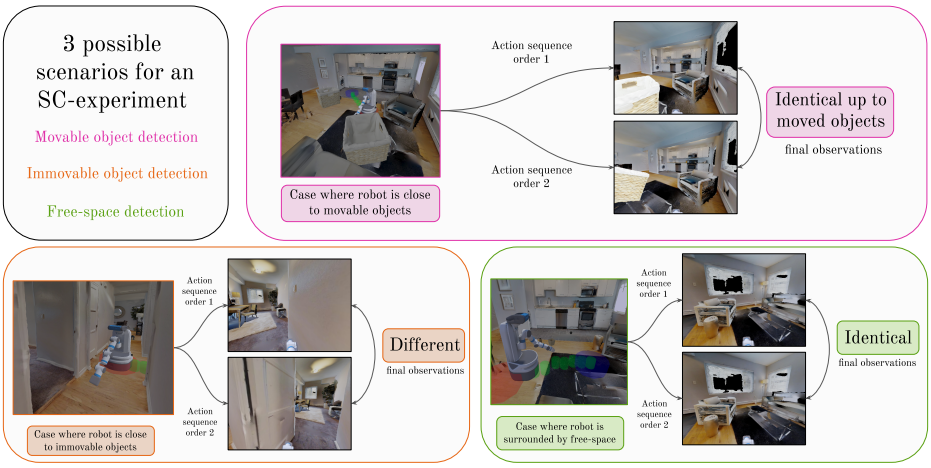


Figure 6.1: Intuition for our approach SCOD for object discovery. 3 scenarios are possible after a SC-experiment (i.e. playing an action sequence in two different orders from the same starting point), depending on the surroundings of the agent. If the agent is surrounded by free-space then the two final observations obs_1 and obs_2 will be identical, whereas if it is surrounded by immovable objects (walls, sofa) then obs_1 and obs_2 will probably be different (because of the different interactions with the immovable objects). If the agent is surrounded by movable objects, obs_1 and obs_2 will be identical up to moved objects that have been interacted with in different manners in the two action sequences.

6.1 SCOD: Object Detection using Sensory Commutativity

6.1.1 Introduction

The role of active movement in object discovery is crucial in the cognitive development of children [36, 76] and is considered a key aspect in theories of perception [53]. Children and animals gradually learn about the objects of the environment, relying on basic mechanisms such as eye movement and motor babbling [5].

We will now present an object discovery method for embodied agents termed SCOD (Sensory Commutativity Object Discovery), based on the analysis of SC-experiment results.

We assess commutativity properties by comparing the two final observations (obs_1, obs_2) obtained after each sequence of the SC-experiment. Based on previous work that study sensory commutativity [133, 131], we posit that there are three potential outcomes when comparing obs_1 and obs_2: they are either completely different, identical, or identical up to moved objects (these scenarios are illustrated in Fig. 6.1):

- obs_1 and obs_2 are different: the two action sequences from this starting position do not commute, because the robot interacted with immovable objects. Consider for instance the robot in a still stance with its arm straight such that the robot stands with a wall at its right. Rotating the base of the robot to the left then to the right would end up with observation obs_1, which is the same observation as the starting situation. Now if the robot rotates its base right then left, since it's blocked by the wall from trying to right first, the robot will end up left to where it started, and it will observe $obs_2 \neq obs_1$. Using the position of the agent, we can now map immovable objects in the environment.

- obs_1 and obs_2 are identical: the two action sequences from this starting position commute, because the robot did not interact with anything in the environment (free movement). An example would be the same situation as in the last paragraph, but with a starting position where the robot is not next to a wall, and stands in a place where there is free space. Rotating left then right, or right then left yields the same observations $obs_1 = obs_2$. Using the position of the agent, we know that there are no objects in the current space around the robot.

- obs_1 and obs_2 are identical except for an object that has been moved: it's the case where the robot has interacted with a movable object that did not block the robot's movement. An example would be having the robot with a movable object to its right in its sight. Rotating left then right would leave no changes in observations, while rotating right then left would push the object out of its sight. Hence, the two action sequences did commute, except for the object that has been moved. We can learn to detect this moving object and track it.

We therefore posit that from these outcomes, the robot can discover and map immovable and movable objects in the environment.

We provide the agent with a basic ability: being able to compare two images. The agent acquires this skill in a pre-training phase where a mask predictor is learned, whose architecture is based on optical flow prediction. Studies in cognitive science indicate that children are capable of doing this differentiation at a very young age (1 month old) [81, 76], so equipping the naive agent with this basic ability is a reasonable assumption. The mask predictor takes two images as input, and outputs two masks corresponding to what has moved between the two observations. Combining the mask predictor and the SC-experiments allows the agents to discover immovable and movable objects in their surroundings. The intuition behind the approach is presented in Fig. 6.2 and on the supplementary video[1].

We use the iGibson interactive environment [155] for simulating this embodied agent scenario. We control the 10-DOF robot Fetch [174], in

[1] `https://youtu.be/Bc5fwZH-CQU`

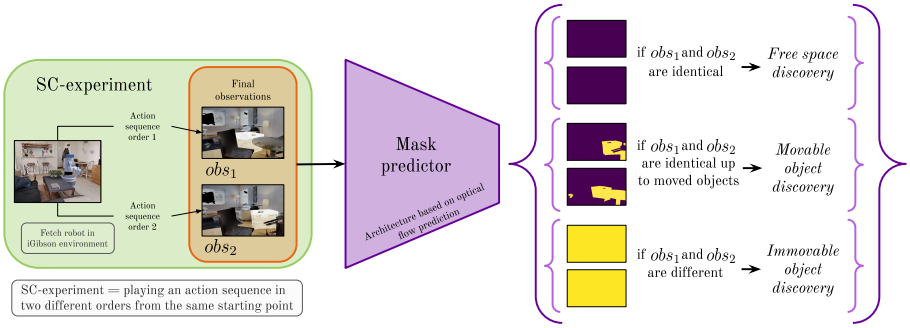


Figure 6.2: Overview of our approach SCOD for object discovery. The agent plays an action sequence in two different orders from the same starting point (SC-experiment). From the two resulting final observations (obs_1, obs_2), we aim at learning about objects in the surroundings of the agent, see Fig. 6.1 for the reasoning behind this idea. For that, a mask predictor is pretrained on procedurally generated data, and applied on (obs_1, obs_2). The mask predictor outputs two binary "difference" masks that represent the semantic difference between (obs_1, obs_2): either identical (all zeros), completely different (all ones) or identical up to moved objects (segmentation masks).

iGibson's 3D scenes reconstructed from real homes. Our results qualitatively and quantitatively showcase the accuracy and generalization properties of SCOD. We also provide a review on object detection and discovery, and compare our work to current state-of-the-art object detection methods, video object segmentation and tracking methods and methods from the robotics literature.

Our contributions are the following:

- We propose SCOD, a novel object discovery method based on sensory commutativity of action sequences.
- We demonstrate the accuracy and generalization properties of SCOD on 3D realistic robotic setups by using the Fetch robot in the iGibson interactive simulator. We also provide real-life generalization examples.
- We compare SCOD to the current landscape of object detection methods. We provide analysis and comparisons to better understand how novel the approach is.

6.1.2 Related work

Passive object detection

Object detection on still images. Passive object detection methods rely on largely annotated databases of object classes that allow to train fast and accurate predictive models for bounding boxes and segmentation masks. Pascal VOC [44] and COCO [101] are examples of the most used datasets, and popular methods like Mask R-CNN [65] or YOLO [141] and their variants have shown to be very efficient at solving object detection and segmentation tasks.

These models are by far the most used in practice: they are extremely useful for a number of real-life applications. Yet, these methods are not suited for open-ended object detection for an agent. The methods are not object-agnostic: they detect objects based on their similarity to objects seen during training. Objects exist in various shapes, colors and sizes in open-world exploration, and while sufficiently large datasets might do the trick, this design does not seem suited for this type of experience.

Video object segmentation and tracking (VOST). VOST aims at expanding the former methods to video data rather than still images. With that comes a number of challenges, such as occlusion, deformation, motion blur, and scale variation. Similarly, researchers have developed strong benchmarks such as DAVIS [130] and Youtube-VOS [180]. The tasks can be summarized as follows: given an input video containing multiple objects, each pixel has to be uniquely assigned to a specific object instance or to the background.

Most well performing methods that have been developed, such as UnOVOST [107] or MOTS [170], rely on propagating the mask of the first frame, which is obtained via passive object detection methods. Thus, the issues mentioned in the previous section also apply to most VOST methods. Few VOST methods try to move away from this paradigm, like STEm-Seg [4], however they still rely on largely annotated datasets, which lead to the same issues mentioned before.

Active object detection

In robotics, there is a large body of work on active object detection (AOD) [14], which is closer to SCOD. The goal is generally to perform active movements, such as poking and pushing in order to learn about movable objects. AOD methods are either based on fix viewpoint or first-person viewpoint.

AOD with fixed viewpoint. In the fix viewpoint scenario, we usually find a robotic arm (without a body or head) in front of a table with multiple objects to interact with [149, 59, 109, 54, 42]. This particular setup is the most common in AOD, as it has many real-life applications (in logistics for instance). However, it comes with constraints that do not apply to

embodied agents, who face first-person viewpoints with partial observability of scenes.

AOD with first-person viewpoint. This setup is the closest to ours. The most common strategy is for the agent to push an object and then update its knowledge of movable objects [11, 148]. This movement is complex for a robot with multiple DOF, which is why this movement is pre-programmed in those methods. In SCOD, we propose an alternative that only relies on random movements and does not make assumptions about available manipulation movements. Also, some methods do not use RGB as input for the object detection methods, but instead 3D cameras [179]. This enhanced input allows to perform object detection in real time, which might be useful for application but does not resolve the problem of learning visual perception from RGB images.

6.1.3 Object discovery method

In order to verify the intuition presented in the introduction, the robot needs to be able to perform an SC-experiment and then detect: 1) if the two resulting observations are identical or not, 2) if they are identical except for the parts of an image corresponding to an object that moved. For that, we equip the agent with a vision system that gets two observations as input and outputs two masks which will be all zeros if the two observations are identical, all ones if they are different, and the mask of an object if this object moves.

Mask predictor training. We thus train a neural network (whose architecture is discussed in the next section) with generated data to predict those two masks with two observations as input. We refer to this model as the "mask predictor". The data to train this model is collected by starting at a random position in the environment (observation obs_1) and then collecting data for the three possible outcomes.

- no difference: it suffices to keep the same observation and the corresponding masks are all zeros. The data is (obs_1 + all zeros mask, obs_1 + all zeros mask).

- completely different: we move the robot and get a different observation obs_2, the corresponding masks are all ones. The data is (obs_1 + all ones mask, obs_2 + all ones mask).

- no difference except moved objects: we randomly disturb the orientation and position of some movable objects and get a new observation obs_2 identical to obs_1 up the moved objects. The data is (obs_1 + moving objects mask, obs_2 + moving objects mask)

The resulting dataset is illustrated in Fig. 6.3.

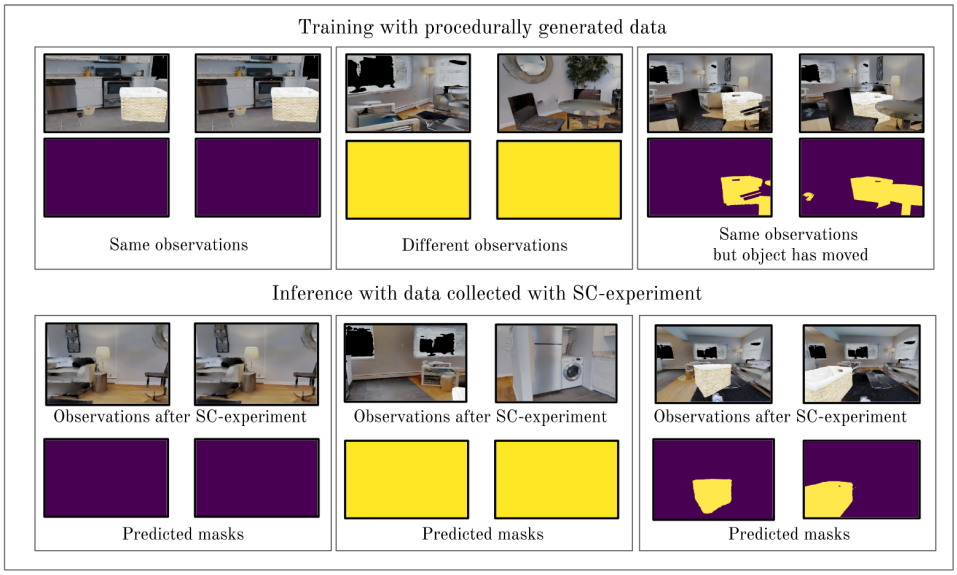


Figure 6.3: **Top:** dataset for training the mask predictor. **Bottom:** inference results on data collected with SC-experiments (each image is the result of one action sequence). The dataset is procedurally generated to simulate the three possible scenarii resulting from a SC-experiment. **Left:** scenario where there are no changes in the observations. **Middle:** scenario where the observations are different. **Right:** scenario where the observations are identical up to moved objects.

Object discovery inference. Once the mask predictor is trained, we place the agent in a random position in the environment and perform SC-experiments where we let it play an action sequence in different orders from the same starting point. Then, the goal is for the agent to detect immovable and movable objects using the generated data from the SC-experiments and the mask predictor.

To summarize, our object discovery method SCOD is divided in two steps:

- Step 1: Train the mask predictor on procedurally generated data.
- Step 2: Discover movable and immovable objects by letting the agent perform SC-experiments and use the trained mask predictor on the resulting observations.

Figure 6.4: Generalization study of movable object detection using SCOD with a mask predictor trained in the Placida environment. In all scenarios, our method correctly predicts the mask object. **Upper left:** Object and environment not seen during training. **Lower left:** Object, environment and field of view not seen during training. **Upper and lower right:** Field of view not seen during training.

6.1.4 Experimental setup

Simulation and environment

We again use the iGibson interactive environment for simulating this embodied agent scenario. In our experiments, we control the Fetch robot [174] equipped with a 7-DOF articulated arm, a base with two wheels, a liftable torso and a RGB camera in its eye.

Mask predictor training (step 1)

Architecture choice reasoning. Predicting the masks given the observations is a process similar to predicting the optical flow of two consecutive frames in a video. In this problem, the state-of-the-art neural network architecture predicts the optical flow field $(2 * W * H)$ using two consecutive RGB frames of a video as input $(2 * 3 * W * H)$. The optical flow field is a projection of the motion field, i.e. the real world 3D motion between the two frames. Thus, the optical flow field corresponds to the displacement of pixels between the two frames. This type of architecture is adapted to our problem since we aim at predicting two binary masks $(2 * W * H)$, using the two final observations (obs_1, obs_2) from the SC-experiment as input

$(2 * 3 * W * H)$. The mask predictor estimates the displacement of objects between the two frames, a similar goal as in optical flow prediction.

For selecting the architecture, we first tested the FlowNet-S architecture [48], a popular baseline for optical flow prediction, as a proof of concept. We then adopted the state-of-the-art RAFT model [163], which performed better. We provide comparisons of the two model performances in the experiments.

Datasets. For training the mask predictor, we procedurally generate 40k training data in the format of tuples $(obs_1, obs_2, mask_1, mask_2)$, as described in Sec.6.1.3. We use the Placida environment, augmented with 40 objects from the YCB object benchmark [21].

Training. For both the Flownet and RAFT model, we train the models using the same architecture and optimization process as proposed by their authors, except for the loss function and the output activation function. We change the loss function to a binary cross-entropy loss between the ground truth mask and the output mask of the network. We select the sigmoid function as output activation function so that the model outputs binary masks instead of the original optical flow map output $(2 * W * H)$. All training details are available in the original open-source implementations we used[2].

Object discovery using SC-experiments (step 2)

SC-experiments. The second part of our object discovery pipeline is to compute SC-experiments and use the trained mask predictor. For the SC-experiments, we play an action sequence in two different orders from the same starting point. We do this by resetting the environment between two action sequences.

To illustrate our results, the action sequences we consider are composed of random (sign, amplitude) motor commands for the DOF of the arm that is closest to the body of the agent. Each action is applied for $\frac{1}{10}$th of a second, and the length of action sequences is set to 20. Note that the choice of DOF is arbitrary, and any other DOF would have worked also. Yet, for illustrative purposes, this DOF empirically allows a well balanced mix of all the three possible scenarios of the SC-experiments.

We test our object discovery method in the Placida environment, as well as different environments not seen during training, such as the Bolton or RS environment. Similarly, in the object discovery step, the objects to detect are not necessarily seen during training, and come from the YCB object benchmark [21].

Evaluation. For qualitative and quantitative evaluation, we manually create a test set with 50 tuples $(obs_1, obs_2, mask_1, mask_2)$ of the three

[2]Link to RAFT and link to FlowNet implementations

possible scenarios resulting from a SC-experiment. We cannot construct this dataset automatically, as the mask has to be manually created by either assessing if the two observations are different or identifying which object has moved between the two observations. Using this dataset, we can first assess the prediction accuracy among the three possible outcomes of a SC-experiment.

In the case where an object has moved (see example in lower right corner of Fig. 6.3), we can further analyze the accuracy of the predicted mask using the Jaccard index, or Intersection over Union (IoU), which is usually used in object segmentation literature [4]. It quantifies the overlap between predicted $(p1, p2)$ and ground-truth (gt_1, gt_2) masks. It is defined as:

$$IoU_i = \frac{|p_i \cap gt_i|}{|p_i \cup gt_i|}$$

.

6.1.5 Results

Our results are best illustrated in video, which is available online[3]. We now present a qualitative and quantitative evaluation of the results.

Qualitative results

Can SCOD detect movable objects? Results presented in Fig. 6.3 & 6.4 illustrate that using the mask detector with the outcome of these SC-experiments can detect objects that have been moved. Note that the mask detector only detects objects that have moved between the two resulting observations, rightfully ignoring the other potential objects that were not moved.

Can SCOD detect immovable objects and free spaces? Results presented in Fig. 6.3 illustrate that the mask predictor is also able to accurately predict when the observations are different or identical. By isolating those two cases from the case where only one or a few objects have moved, we can map the starting position of the agent with the probability that an SC-experiment will commute. In Fig. 6.5, each dot represents a starting position, and the dot's color is the probability of observing the outcome "different" when playing a SC-experiment at this position (darker is higher). Regions with dark dots correspond to regions where there are walls and immovable objects in the way of Fetch's arm, whereas regions with white dots correspond to free spaces.

Indeed, in the kitchen part (room at the top), the space is cramped and so most of the positions indicate low commutation probability (less than 0.4) because of the interactions induced with the furniture. In the living room

[3] `https://youtu.be/Bc5fwZH-CQU`

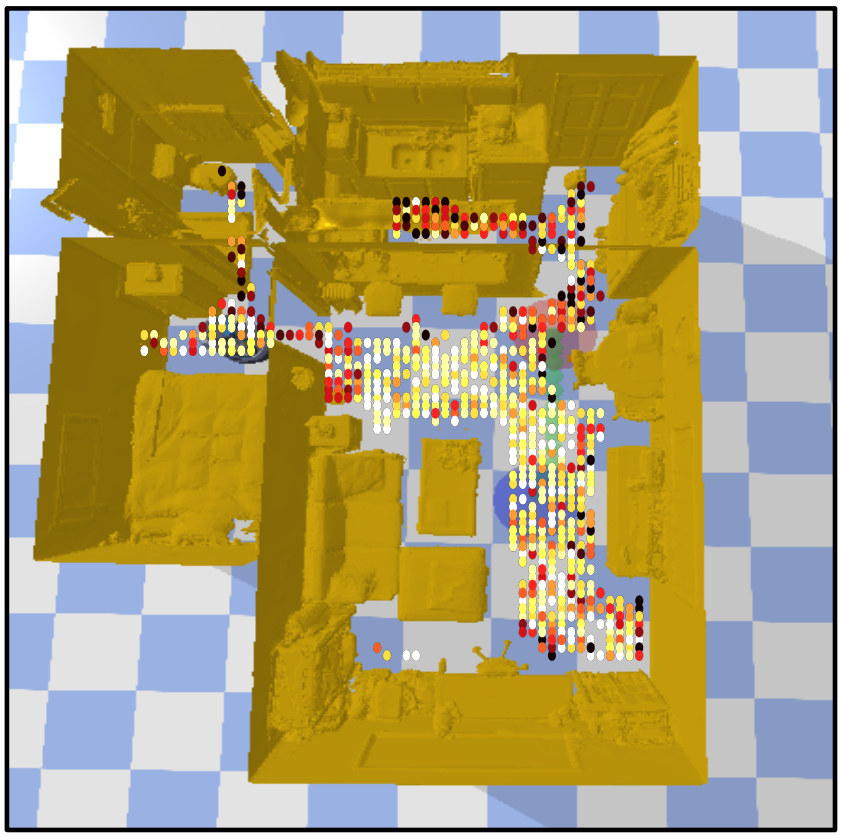


Figure 6.5: Immovable object detection using SCOD. Each dot represents the probability of observing the outcome "different" when playing a SC-experiment at this position (darker is higher). Free spaces are filled with white dots and cramped spaces with darker dots.

(main room) and the bedroom (at the left), most empty spaces show high probability (around 0.8 and 1.0). We thus obtain a mapping of immovable objects and free spaces using SCOD predictions.

Object tracking after detection. After movable object detection, we can then use semi-supervised tracking algorithms in order to track the detected object. Fig.6.6 illustrates the detection and tracking pipeline using SCOD and Space-Time memory networks (STM) [121]. We predict the first mask using SCOD, and then track the detected object using STM. We obtain a quasi-perfect automatic tracking of the detected object.

Algorithm design alternatives

We question design choices in the SCOD algorithm. First, we question the use of SC-experiments compared to simpler alternatives such as just playing

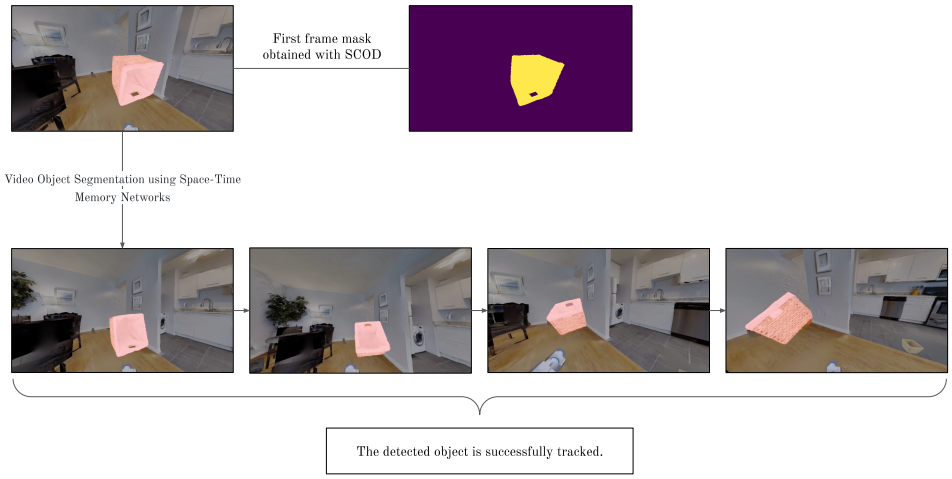


Figure 6.6: Object detection and tracking pipeline. We first use SCOD to detect an object, and use the learned mask to track it using STM, a semi-supervised video object segmentation algorithm.

an action sequence and comparing the first and last observations. This method would rarely detect objects because most experiments would result in a complete image change where the SC-experiments would highlight only a particular object, as illustrated in Fig. 6.7. Another alternative would be to start in a position, play an action sequence, and then go back to this starting point and compare what's changed. While this approach would be comparable for movable object detection, this would not allow detecting immovable objects and free-space.

Second, an alternative to the use of a mask predictor would be to use a naive image subtraction between the two observations resulting from the SC-experiment. However, if one object has moved, the naive image subtraction results in two masks (one for each position of the object that has moved). These masks can overlap, and thus be hard to distinguish. Then if more than one object has moved, the subtraction will prove difficult to interpret. This issue is illustrated in Fig. 6.7. We initially tried this, and then switched to the mask predictor which ended up being more efficient.

Quantitative results

We present the quantitative results on the manually collected test set in Tab. 6.1 (see non-generalization test set column). With the RAFT architecture for the mask predictor, we reach an average Jaccard index of 0.97 for movable object detection and respectively 90.4% and 96.1% for immovable object and freespace detection. These results highlight the efficiency of SCOD for all three tasks.

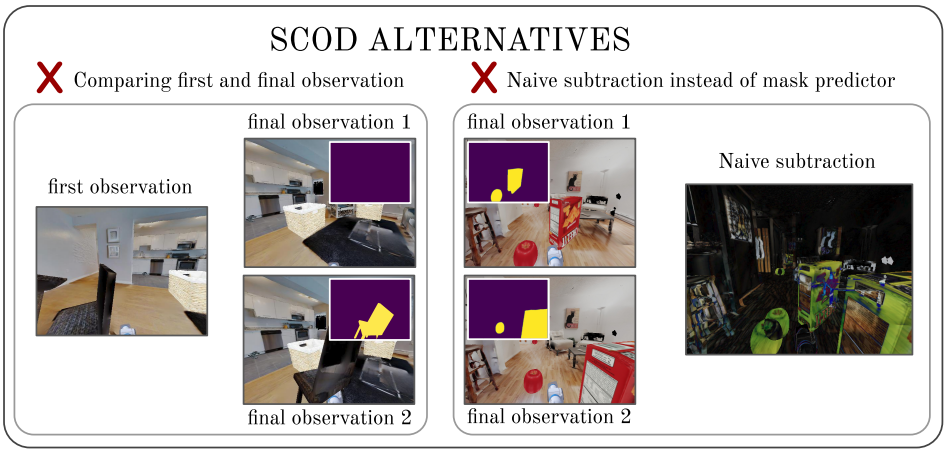


Figure 6.7: Algorithm design alternatives for SCOD. Comparing first and last observations fails, or replacing the mask predictor by naive subtraction are not viable options, which justifies the use of SC-experiments and mask predictor in SCOD. For comparison we provide the masks computed by SCOD over the final observations.

Table 6.1: Quantitative results for movable objects, immovable objects and free space detection using Flownet and RAFT architecture for the mask predictor. Both methods are tested on a test set for training, and on a test set for generalization to unseen environments and objects.

	No generalization test set			Generalization test set		
Mask predictor	Movable object	Immovable object	Free space	Movable object	Immovable object	Free space
Flownet	0.86 (IoU)	61.9% (acc.)	95.8% (acc.)	0.61 (IoU)	75.0% (acc.)	99.3% (acc.)
RAFT	**0.97** (IoU)	**90.4%** (acc.)	96.1% (acc.)	**0.84** (IoU)	**90.9%** (acc.)	98.6% (acc.)

Regarding the choice of mask predictor architecture, RAFT overall performance is significantly better than FlowNet. This justifies the adoption of RAFT as the mask predictor. In Fig.6.8, we provide a qualitative comparison between object mask prediction of Flownet and RAFT. Both models are able to detect the moved object, hence they are suited for the SCOD method. However, the quantitative gap shown in Tab.6.1 is explained by the higher accuracy of RAFT over Flownet. The exact shape of the objects is not perfectly captured by Flownet. On the contrary, RAFT is often able to predict the exact shape of the moved objects.

Generalization study

Does SCOD generalize to unseen environments and objects? In principle, this movable and immovable object detection method is designed

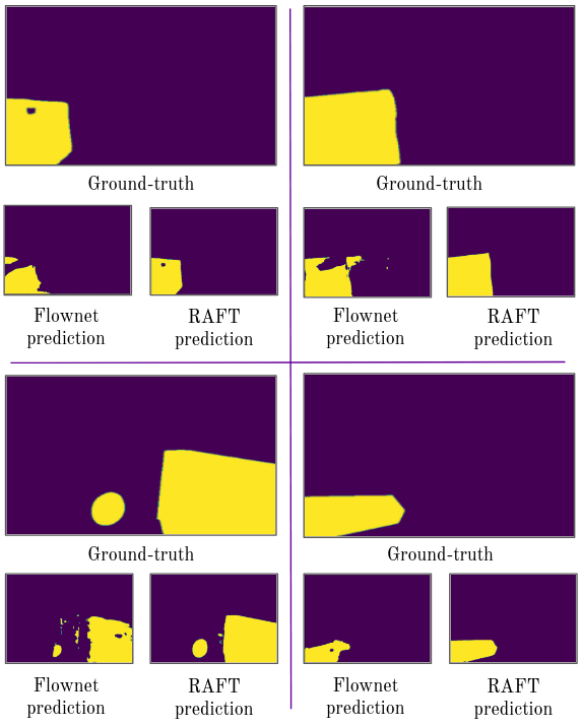


Figure 6.8: Comparison of mask predictions between Flownet and RAFT. While Flownet is able to roughly predict the object mask, RAFT has a better accuracy.

to work in any environment, any objects and any field of view. Indeed, it only relies on having a precise mask predictor, which we show can be achieved. We thus performed a generalization study of our method. We manually created a generalization test set (150 instances) with data consisting of objects, environments and field of view that were not shown during training. For this study, we selected the Bolton environment, 20 objects from the YCB benchmark that were not shown during training, and a bigger field of view (90 versus 45 for training).

In Fig. 6.4 and Tab. 6.1, we show results for the generalization study, which indicate that the mask predictor can indeed be used with environments, objects, and field of view that have been not shown during training.

Qualitatively, the mask predictor is able to precisely predict which objects have moved, regardless of the shape of the object, and of the nature of the background and field of view. Quantitatively, the precision of SCOD shows strong generalization. We reach an average Jaccard index of 0.84 for movable object detection. For immovable object detection and freespace detection, the performance is similar between the generalization and non-generalization test set. Hence, the low performance drop between

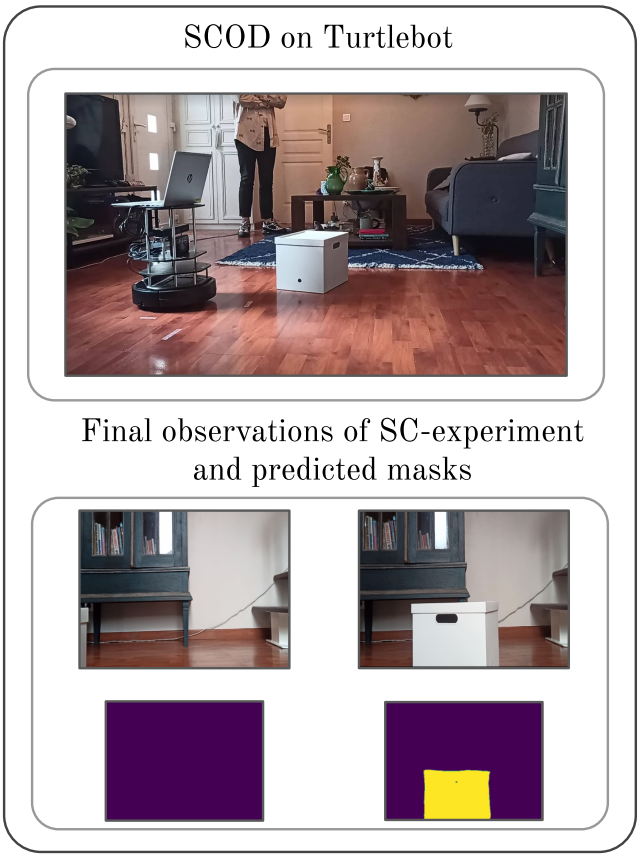


Figure 6.9: Object detection on Turtlebot using SCOD: the robot performed a SC-experiment which led to a moved object that is detected by SCOD, see supplementary video. We present the two resulting observations from the SC-experiment, and the predicted masks below. The algorithm has solely been trained on synthetic images, and generalizes to real-life scenarios.

in-distribution and out-of-distribution test sets allows us to conclude that SCOD generalizes to new environments, objects and fields of view.

Real robot generalization. As a last generalization test, we performed SC-experiments in real-life by using a Turtlebot robot. Qualitative results are presented on Fig. 6.9 and illustrated on the supplementary video[4]. We use the same mask predictor, which has been trained on synthetic images solely, and we obtain satisfying qualitative results. The methods seem able to bridge the reality gap.

[4] `https://youtu.be/Bc5fwZH-CQU`

6.1.6 Discussion and conclusion

We deployed SC-experiments in real-life with Turtlebot as a demonstration, but there are a few difficulties for a more advanced real-life deployment of SCOD. We need the agent to play two action sequences from the same starting point. In real-life, the method has to overcome stochasticity and irreversible actions (e.g. breaking a glass) which break that assumption. Also, if an object is moved, you would have to place it back to its original position.

However, this could be overcomed by learning an accurate forward model of the environment that allows the agent to predict what will happen when it plays an action sequence. The forward model would act as a proxy for one of the sequences, and the robot would perform the other sequence in real life, therefore performing SC-experiments by comparing real experience with imagination. Recent works have made significant progress in this direction [60, 63]. We believe this is an important future work for using sensory commutativity to build perception for artificial agents.

6.2 Sensory Commutativity for efficient RL

We now illustrate how SCP can be used for unsupervised exploration, by using it to improve sample-efficiency in an RL setup. For computational reasons, we experiment with the Flatland simulator.

6.2.1 Experimental setup

We use the PPO2 [151] implementation from Stable-Baselines [71]. The policy is composed of a 1D convolutional feature extractor followed by a recurrent policy. We consider the same agent, Polyphemus, for which we computed the SCP criterion in Fig. 5.4. The input of the policy is the RGB image of what Polyphemus' eye sees. The environment considered is a square room with 3 dead zones (which terminate the episode with a -20 reward) and a goal zone (which terminates the episode with a +50 reward), illustrated in Fig. 6.10. We propose two methods that take advantage of the SCP to modify the action space of the agent. The goal is to improve sample-efficiency when learning to solve a task in this embodied scenario.

SCP-truncated action space. We propose to focus exploration on the degrees of freedom that have a high impact on the environment, by fixating degrees of freedom corresponding to high SCP. We implement this by halving the dimension of the action space, keeping only the degrees of freedom that have the most effect on the environment, i.e. lower SCP value. We thus keep the base movement and rotation, and the shoulders joint, while discarding the elbow joints, head rotation, and eyelid activation. We refer to this method as *SCP-truncated* action space. This action space reduction will

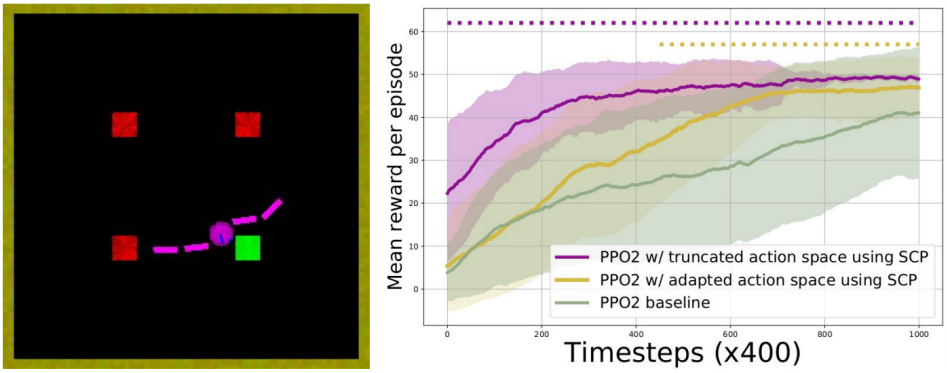


Figure 6.10: **Left:** RL task. **Right:** Results.

simplify the RL task, as long as the necessary actions such as base motion are selected by the SCP criteria.

SCP-adapted action space. A less involved proposition is to modify the action sampling interval according to the SCP value, for each degree of freedom. This method will modify the exploration dynamics to favor important actions. Suppose that the sampling interval for each dimension of the action space is $[-1, 1]$. If a dimension has high SCP, i.e. it does not affect the environment a lot, we then reduce the interval from which actions are sampled $[-1 \cdot l(SCP), 1 \cdot l(SCP)]$. The function l maps the highest SCP to 0 and lowest SCP to 1, then we use a linear interpolation between those two points to deduce values for $SCP \in]-1, 1[$. We refer to this method as *SCP-adapted* action space.

Comparison protocol. We compare those two strategies to a baseline policy trained to solve the task with the complete action space. We average the result of each policy over 30 trials initialized with different random seeds, and we test the statistical significance of our results according to the guidelines provided by [34].

6.2.2 Results

The results are displayed on Fig. 6.10 (right). First, we notice that all strategies are viable to solve the task. We now compare sample-efficiency between the strategies. The policy trained with *SCP-truncated* action space can learn how to solve the task more than twice as fast as the baseline policy. The discarded degrees of freedom are not crucial in this navigation task, hence the agent is still able to solve the task using only the degrees of freedom that have the lowest SCP value. The policy trained with *SCP-adapted* action space is less sample-effective than the *SCP-truncated* but still learns significantly faster than the baseline policy.

6.3 Conclusion

In this Chapter, we proposed and evaluated two algorithms based on the properties of Sensory Commutativity. We showed that SCP allows to train RL agents faster by truncating or adapting the action space using the SCP criterion.

We also presented SCOD, an object detection method for embodied agents based on an analysis of the potential outcomes of a SC-experiment. By using architectures based on optical flow prediction, SCOD permits an active object detection based on random exploration of the environment.

Chapter 7

Conclusion and perspectives

Contents

7.1 Conclusion

The aim of this thesis was to study agent perception using modern Machine-Learning methods.

We first defined in Chapter 1 the problem of agent perception learning, and drew the different relations with the current state-of-the-art of ML research. The problem we consider is an embodied agent equipped with first person sensors and a body with several DOF that allow navigation and manipulation. The environments we consider are realistic looking, share as many features with the real world as possible: physics laws, temporal continuity, different sources of illuminations, etc.

In Chapter 2, we defined the problem of perception which is studied in this thesis, and introduced the hypothesis that we use as the basis to our work. We reviewed the most common experimental setups: robots and simulators, and assessed their features regarding the question of agent perception as we defined it.

We then went over the sub-fields of ML that research the question of agent perception in Chapter 3. They are all related to Representation Learning: the hierarchical learning of representations of data. We have approaches that aim at building a model of the environment from the point of view of the agent, we have RL that aims at learning about the world by trying to solve tasks, we have Continual Learning that aims at learning over extended periods of time where the data distribution can evolve. We also presented our contributions to those sub-fields: S-TRIGGER (a Continual State Representation Learning method that uses generative models and an automatic environment change detection method) and DisCoRL (a Continual Reinforcement Learning method that can be deployed on real robots). We continued in Chapter 4 by studying a crucial aspect of Representation Learning: disentanglement. We investigated a recently proposed approach for disentanglement and provided theoretical and practical contributions for applying this method of disentanglement.

Chapters 5 and 6 presented our contributions in a novel approach for agent perception: the study of sensory commutativity of action sequences. We presented theoretical motivations and contributions for this work. We then illustrated the usefulness of this study by providing a method for automatic characterization of body degrees of freedom called SCP (Sensory Commutativity Probability), and an active object detection method called SCOD (Sensory Commutativity Object Detection).

We hope that this manuscript and our contributions can help progress on the long-standing quest of having agents that perceive our world as we do, and can help us live better lives.

7.2 Perspectives and discussion

We discuss the perspectives of the approaches presented in this thesis for the problem of perception. We shed light on potential interesting future work.

7.2.1 State Representation Learning

The large field of Representation Learning is evolving very fast, as novel representation learning methods are invented everyday. These methods can often be applied to State Representation Learning directly. The contributions we made in SRL are based on the representation learning models that were available at the time. Hence, progress in Representation Learning often translates into progress in SRL.

Regarding generative models, tremendous progress has happened since VAEs and GANs have been proposed and developed in the early 2010s. Back then, having a generative model to sample high-resolution images was impossible and today these models are common. Stylegan 1 [79] and 2 [80]

(and the recent Stylegan2-ADA variant [78]) allows to generate photorealistic images of high resolution (1024*1024 pixels) with a dataset of a few thousands images. These models build on the improvements made to the GAN loss function for improved stability of training, as well as scaling in the number of parameters and general architecture to allow better sampling quality. VQ-VAEs (VQ-VAE [122] and VQ-VAE2 [139]) [64] are the most recent development of VAEs and also allow to match the quality of generation seen with the best GANs (but with the additional encoding ability that comes with VAEs). They use discrete representation of data instead of continuous ones, which allows them to scale to high resolution images without loss of sampling quality. Transformers are also recently making their way in the realm of images, departing from the sole application to NLP [40, 127], with exceptional results and SOTA performances. The challenge in applying Transformers to images is avoiding the quadratic nature of the attention operation, which is usually circumvented using sparse or localized attention to reduce the computational overload, just like going from densely connected neural networks to convolutions. Their general way of learning on text and images even allows modeling heterogeneous data jointly. DALL-E [138] is one example of this: a transformer based architecture is used for generating images based on a text and image prompt, with promising results.

A promising approach for research in State Representation Learning is to apply those novel representation learning models to agent-environment setups rather than still datasets for which the method was designed, which already showed some interesting results [126]. As these models get better we hope to also progress in agent perception, and eventually come closer to having agents that have general and transferable knowledge in a wide range of environments.

7.2.2 Reinforcement Learning

Model-free reinforcement learning has been the most popular approach in the 2010s, and we are now seeing the limits of such approaches because it is not easy to learn general and transferable knowledge in this setup. This is why we are seeing more and more model-based approaches, which aim at learning a model of the world and then use it to solve tasks with planning algorithms [63].

Other approaches are getting traction like Language-Conditioned RL [31, 172], where the agent gets additional signals in the form of natural language or symbolic feedback. This allows them to depart from the setup where the agent has the reward as the sole signal for learning, which is one the reason RL has not been sample-efficient enough to be applied to real-life scenarios easily. Another approach that seems promising for RL is the multi-task/multi-goal setup [35], where the agent can learn to solve a large number of tasks using the same experience. This is another form of augmenting the

information in the learning signal that the algorithm uses. By doing so, RL agents learn more robust knowledge about their environment and thus improve their perception.

Still, research in RL is very active at the moment, and we can expect to see novel breakthroughs in the near future. In DisCoRL, these novel algorithms could be applied to speed the process of learning.

7.2.3 Continual Learning

Progress in CL is important for the problem of perception since novel methods of CL developed could be used and adapted to this agent-environment scenario. The long term goal is being able to produce an agent that continuously learns in an autonomous manner. If the agent can selectively forget non-crucial skills and knowledge, while still evolving its acquired skills, this would be great help toward the development of autonomous agents that act and learn in the real-world.

While the Continual Learning community is growing fast, it is hard to assess the progress that has been made over the last years, because of a lack of standard benchmark. There is a current effort to classify and organize all the different tasks and learning scenarios in CL, whether it is in the RL setup [83] or in the supervised/unsupervised setup [94]. The supervised learning setup gets the most progress currently [62, 20, 75], but there are worrying signs that might show that progress is not as fast as we might consider it is. For instance, we have the GDumb paper [135], that shows some benchmarks are too oversimplified, which leads to seeing naive baselines as good as state-of-the-art algorithms. GDumb simply greedily stores samples in memory as they come and; at test time, trains a model from scratch using samples only in the memory. The fact that these baseline models perform so well on commonly accepted benchmarks does show that there is a long way to go before having robust benchmarks for CL.

However, with its large community and strong presence in tier-1 conferences workshops and main track, CL is inexorably getting more and more attention which accelerates progress in the field. 3

Better CL methods means that methods like DisCoRL or S-TRIGGER can be improved. Those methods introduce a continual learning concept, and then the experiments are based on a particular instantiation of the algorithms available at the time of the experiments. It would be interesting to evaluate how these methods can perform using novel generative models, novel continual learning algorithms, etc.

7.2.4 Perception theories combined with contemporary ML

There is an ongoing effort of applying interdisciplinary research to the question of embodied perception, by bringing together research in Computer Sci-

ence, Cognitive Science, Psychology, Brain Science, Developmental Robotics and various other related fields. This effort aims at answering important questions that become more deeply studied in the field of Machine Learning: How far is the state-of-the-art machine intelligence from babies? How does a baby learn from their own interactions and experiences? What sort of insights can we acquire from the baby's mind? How can those insights help us build smart machines with baby-like intelligence?

Because human babies gradually make sense of the environment through their experiences, a process known as learning by doing, this approach might avoid the problem of requiring a vast amount of labeled data to train ML algorithms. Biological agents actively engage with their surroundings and explore the world through their own interactions. They gradually acquire the abstract concept of objects and develop the ability to generalize problems. Thus, if we understand how a biological agent's mind develops, we can imitate those learning processes in machines and thereby solve previously unsolved problems such as domain generalization and overcoming the stability-plasticity dilemma.

In this field we have several sub-fields such as neuro-cognitive learning systems, curiosity-driven self-supervised learning, Active learning based on a perception-cognition-action cycle, Embodied concept learning from real-world interactions, object-centric representation and concept learning, developmental / cognitive robotics or Social and emotional development systems. Each of these subfields take inspiration from perception theories in an interdisciplinary effort to make progress on the question of artificial perception.

7.2.5 Sensory Commutativity

In this thesis, we formally introduced the notion of Sensory Commutativity of action sequences and provided algorithms that show how to use the concept to learn about the agent and the environment.

Sensory Commutativity-based algorithms can be improved in several ways. First, the core component of the algorithms proposed in our contributions is the SC-experiment. While the SC-experiment is feasible in simulation, it is still a challenge to perform it efficiently in real-life. For now, a human experimenter would have to reset the scene between two trajectories. Future work can improve on the implementation of such experiments in controlled settings, or with the use of forward models to avoid the resetting of objects between the two action sequences. Examples such as [63] were not available at the beginning of our research work, and could be implemented in our setup in order to improve on the real-world deployment of SC-experiments.

Also, the idea of using Sensory Commutativity to learn about the environment and the agent can be extended in multiple ways. We showed how to perform object detection, and learn about the relative importance of the

degrees of freedom of the agent, but the use of SC is surely not limited to this. For instance one could experiment with affordance detection using an extended version of SCOD. Basically SC provides a new learning mechanism for the problem of environment and agent discovery.

Finally, SC could be associated with other learning mechanisms like intrisic curiosity or other clever exploration methods to further enhance the performance and efficiency of SC-based methods.

7.2.6 Robots and simulations

Robots. The development of physical robots has steadily progressed for the past 50 years. It is of course a complex field with different actors: research scientists, and industry conceptors. There are very different incentives that push folks to create robots. In this thesis we are interested in robots that can navigate and solve tasks using their body (e.g. through manipulation) in human environments. So they have to be mobile, and also have finer controls with a robotic arm and fingers for instance. Examples of recently created robots in research environments that satisfy those criteria are the Rollin' Justin robot [50], the TORO robot [43], the Fetch robot [174] (which we use in our simulation experiments in Chapter 6, and illustrate in Fig.7.1).

Figure 7.1: The Fetch robot, which allows navigation with a mobile base, and manipulation with its 10-DOF articulated arm. The robot has first-person embedded sensors (camera).

In the industry, we can cite the Spot robot of Boston Dynamics, which looks promising for navigation, and can even manipulate the environment when it is equipped with an arm, see Fig.7.2. This robot could be a promising testbed for a multi-purpose learning robot. By shipping ML-based methods in the robot, we could see progress towards having real robots learning in the real world, adapting to new situations and solving tasks.

Figure 7.2: The Spot robot from Boston Dynamics, equipped with an articulated arm that allows manipulation. The robot has first-person embedded sensors (camera).

Simulators for embodied perception. Simulators for embodied perception hav developed fast in the recent years. These simulators include iGibson [155] (illustrated on Fig.7.3), AI2Thor [86], AI Habitat [147], Isaac Sim [114], Sapien [178] and TDW [52]. Most of these simulations allow navigation, manipulation and reasoning with embodied agents in realistic environments like indoor or outdoor places. Even though not all simulations have all the required aspects for studying the embodied perception problem (as described in Chapter 2), most of them are pushing in an interesting direction for open-access, reproducibility and benchmarks for the research on perception.

As for benchmarks, a workshop on embodied AI at the Conference on Computer Vision and Pattern Recognition (CVPR) is happening in 2021[1], and illustrates well the tremendous progress being made in the creation of benchmarks for this research area. The competition track offers 13 different tasks, including visual navigation, object rearrangement, embodied question answering, simulation-to-real transfer and embodied vision & language tasks. The simulations used in this workshop are Isaac Sim, iGibson, AI Habitat and AI2-Thor. This effort is promising because it gives a chance for a proper embodied perception benchmark to emerge, where research can compete fairly, since everyone has access to the same simulation. More gen-

[1] `https://embodied-ai.org/`

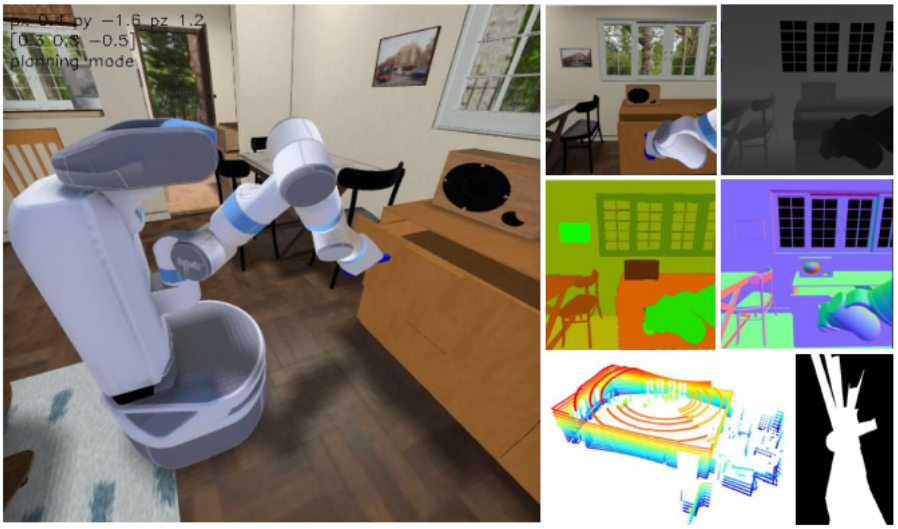


Fig. 3: Robot interacting in iGibson (large picture: 3rd person view)

Figure 7.3: The iGibson simulator is one of the simulators that allows research progress in embodied perception. By using simulated robots that actually exist in real life and realistic indoor environments, the simulation-to-real transfer is easier while allowing researchers to compare themselves easily on the same setup.

erally, the competition, these tasks and simulations in general put a great importance on the embodied aspect. This is in line with the main argument of this thesis on embodied perception so we believe that this effort is promising.

Bibliography

[1] Alessandro Achille, Tom Eccles, Loic Matthey, Christopher P Burgess, Nick Watters, Alexander Lerchner, and Irina Higgins. Life-long disentangled representation learning with cross-domain latent homologies. *arXiv preprint arXiv:1808.06508*, 2018.

[2] Marcin Andrychowicz, Filip Wolski, Alex Ray, Jonas Schneider, Rachel Fong, Peter Welinder, Bob McGrew, Josh Tobin, OpenAI Pieter Abbeel, and Wojciech Zaremba. Hindsight experience replay. In *Advances in neural information processing systems*, pages 5048–5058, 2017.

[3] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein gan. *arXiv preprint arXiv:1701.07875*, 2017.

[4] Ali Athar, Sabarinath Mahadevan, Aljoša Ošep, Laura Leal-Taixé, and Bastian Leibe. Stem-seg: Spatio-temporal embeddings for instance segmentation in videos. *arXiv preprint arXiv:2003.08429*, 2020.

[5] Renee Baillargeon, Elizabeth S Spelke, and Stanley Wasserman. Object permanence in five-month-old infants. *Cognition*, 20(3):191–208, 1985.

[6] Gianluca Baldassarre, Tom Stafford, Marco Mirolli, Peter Redgrave, Richard M Ryan, and Andrew Barto. Intrinsic motivations and open-ended development in animals, humans, and robots: an overview. *Frontiers in psychology*, 5:985, 2014.

[7] Andrew G Barto. Intrinsic motivation and reinforcement learning. In *Intrinsically motivated learning in natural and artificial systems*, pages 17–47. Springer, 2013.

[8] Charles Beattie, Joel Z Leibo, Denis Teplyashin, Tom Ward, Marcus Wainwright, Heinrich Küttler, Andrew Lefrancq, Simon Green, Víctor Valdés, Amir Sadik, et al. Deepmind lab. *arXiv preprint arXiv:1612.03801*, 2016.

[9] Marc G Bellemare, Yavar Naddaf, Joel Veness, and Michael Bowling. The arcade learning environment: An evaluation platform for general agents. *Journal of Artificial Intelligence Research*, 47:253–279, 2013.

[10] Yoshua Bengio, Aaron Courville, and Pascal Vincent. Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence*, 35(8):1798–1828, 2013.

[11] Christian Bersch, Dejan Pangercic, Sarah Osentoski, Karol Hausman, Zoltan-Csaba Marton, Ryohei Ueda, Kei Okada, and Michael Beetz. Segmentation of textured and textureless objects through interactive perception. 2012.

[12] David Berthelot, Thomas Schumm, and Luke Metz. Began: boundary equilibrium generative adversarial networks. *arXiv preprint arXiv:1703.10717*, 2017.

[13] Alexey Bochkovskiy, Chien-Yao Wang, and Hong-Yuan Mark Liao. Yolov4: Optimal speed and accuracy of object detection. *arXiv preprint arXiv:2004.10934*, 2020.

[14] Jeannette Bohg, Karol Hausman, Bharath Sankaran, Oliver Brock, Danica Kragic, Stefan Schaal, and Gaurav S Sukhatme. Interactive perception: Leveraging action in perception and perception in action. *IEEE Transactions on Robotics*, 33(6):1273–1291, 2017.

[15] Leo Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

[16] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym. *arXiv preprint arXiv:1606.01540*, 2016.

[17] Yuri Burda, Harrison Edwards, Amos Storkey, and Oleg Klimov. Exploration by random network distillation. *arXiv preprint arXiv:1810.12894*, 2018.

[18] Christopher JC Burges. A tutorial on support vector machines for pattern recognition. *Data mining and knowledge discovery*, 2(2):121–167, 1998.

[19] Christopher P Burgess, Irina Higgins, Arka Pal, Loic Matthey, Nick Watters, Guillaume Desjardins, and Alexander Lerchner. Understanding disentangling in β-vae. *arXiv preprint arXiv:1804.03599*, 2018.

[20] Massimo Caccia, Pau Rodriguez, Oleksiy Ostapenko, Fabrice Normandin, Min Lin, Lucas Caccia, Issam Laradji, Irina Rish, Alexande Lacoste, David Vazquez, et al. Online fast adaptation and knowledge accumulation: a new approach to continual learning. *arXiv preprint arXiv:2003.05856*, 2020.

[21] Berk Calli, Aaron Walsman, Arjun Singh, Siddhartha Srinivasa, Pieter Abbeel, and Aaron M Dollar. Benchmarking in manipulation research: The ycb object and model set and benchmarking protocols. *arXiv preprint arXiv:1502.03143*, 2015.

[22] Rich Caruana. Multitask learning. *Machine Learning*, 28(1):41–75, Jul 1997.

[23] Hugo Caselles-Dupré, Louis Annabi, Oksana Hagen, Michael Garcia-Ortiz, and David Filliat. Flatland: a lightweight first-person 2-d environment for reinforcement learning. *arXiv preprint arXiv:1809.00510*, 2018.

[24] Hugo Caselles-Dupré, Michael Garcia-Ortiz, and David Filliat. Continual state representation learning for reinforcement learning using generative replay. *arXiv preprint arXiv:1810.03880*, 2018.

[25] Hugo Caselles-Dupré, Michael Garcia-Ortiz, and David Filliat. S-trigger: Continual state representation learning via self-triggered generative replay. *arXiv preprint arXiv:1902.09434*, 2019.

[26] Hugo Caselles-Dupré, Michael Garcia-Ortiz, and David Filliat. Symmetry-based disentangled representation learning requires interaction with environments. *arXiv preprint arXiv:1904.00243*, 2019.

[27] Hugo Caselles-Dupré, Michael Garcia-Ortiz, and David Filliat. Object detection for embodied agents using sensory commutativity of action sequences. *NeurIPS 2020 workshop on BabyMind*, 2020.

[28] Hugo Caselles-Dupré, Michael Garcia-Ortiz, and David Filliat. On the sensory commutativity of action sequences for embodied agents. *arXiv preprint arXiv:2002.05630*, 2020.

[29] Paul Christiano, Zain Shah, Igor Mordatch, Jonas Schneider, Trevor Blackwell, Joshua Tobin, Pieter Abbeel, and Wojciech Zaremba. Transfer from simulation to real world through learning deep inverse dynamics model. *arXiv preprint arXiv:1610.03518*, 2016.

[30] Ching-Yao Chuang, Jiaman Li, Antonio Torralba, and Sanja Fidler. Learning to act properly: Predicting and explaining affordances from images. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 975–983, 2018.

[31] Cédric Colas, Ahmed Akakzia, Pierre-Yves Oudeyer, Mohamed Chetouani, and Olivier Sigaud. Language-conditioned goal generation: a new approach to language grounding for rl. *arXiv preprint arXiv:2006.07043*, 2020.

[32] Cédric Colas, Pierre Fournier, Mohamed Chetouani, Olivier Sigaud, and Pierre-Yves Oudeyer. Curious: intrinsically motivated modular multi-goal reinforcement learning. In *International conference on machine learning*, pages 1331–1340. PMLR, 2019.

[33] Cédric Colas, Olivier Sigaud, and Pierre-Yves Oudeyer. CURIOUS: Intrinsically Motivated Multi-Task, Multi-Goal Reinforcement Learning. *arXiv preprint arXiv:1810.06284*, 2018.

[34] Cédric Colas, Olivier Sigaud, and Pierre-Yves Oudeyer. How many random seeds? statistical power analysis in deep reinforcement learning experiments. *arXiv preprint arXiv:1806.08295*, 2018.

[35] Cédric Colas, Tristan Karch, Olivier Sigaud, and Pierre-Yves Oudeyer. Intrinsically motivated goal-conditioned reinforcement learning: a short survey, 2020.

[36] John Colombo. The development of visual attention in infancy. *Annual review of psychology*, 52(1):337–367, 2001.

[37] Erwin Coumans and Yunfei Bai. Pybullet, a python module for physics simulation for games, robotics and machine learning. 2016.

[38] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[39] Thanh-Toan Do, Anh Nguyen, and Ian Reid. Affordancenet: An end-to-end deep learning approach for object affordance detection. In *2018 IEEE international conference on robotics and automation (ICRA)*, pages 5882–5889. IEEE, 2018.

[40] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

[41] Arthur Douillard and Timothée Lesort. Continuum: Simple management of complex continual learning scenarios. *arXiv preprint arXiv:2102.06253*, 2021.

[42] Andreas Eitel, Nico Hauff, and Wolfram Burgard. Learning to singulate objects using a push proposal network. In *Robotics Research*, pages 405–419. Springer, 2020.

[43] Johannes Englsberger, Alexander Werner, Christian Ott, Bernd Henze, Maximo A Roa, Gianluca Garofalo, Robert Burger, Alexander Beyer, Oliver Eiberger, Korbinian Schmid, et al. Overview of the torque-controlled humanoid robot toro. In *2014 IEEE-RAS International Conference on Humanoid Robots*, pages 916–923. IEEE, 2014.

[44] Mark Everingham, SM Ali Eslami, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. The pascal visual object classes challenge: A retrospective. *International journal of computer vision*, 111(1):98–136, 2015.

[45] Joël Fagot and Robert G Cook. Evidence for large long-term memory capacities in baboons and pigeons and its implications for learning and the evolution of cognition. *Proceedings of the National Academy of Sciences*, 103(46):17564–17567, 2006.

[46] Kuan Fang, Te-Lin Wu, Daniel Yang, Silvio Savarese, and Joseph J Lim. Demo2vec: Reasoning object affordances from online videos. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2139–2147, 2018.

[47] Chrisantha Fernando, Dylan Banarse, Charles Blundell, Yori Zwols, David Ha, Andrei A. Rusu, Alexander Pritzel, and Daan Wierstra. Pathnet: Evolution channels gradient descent in super neural networks. *CoRR*, abs/1701.08734, 2017.

[48] Philipp Fischer, Alexey Dosovitskiy, Eddy Ilg, Philip Häusser, Caner Hazırbaş, Vladimir Golkov, Patrick Van der Smagt, Daniel Cremers, and Thomas Brox. Flownet: Learning optical flow with convolutional networks. *arXiv preprint arXiv:1504.06852*, 2015.

[49] Robert M. French. Catastrophic forgetting in connectionist networks. *Trends in Cognitive Sciences*, 3(4):128–135, 1999.

[50] Matthias Fuchs, Ch Borst, P Robuffo Giordano, Andreas Baumann, Erich Kraemer, Jörg Langwald, Robin Gruber, Nikolaus Seitz, Georg Plank, Klaus Kunze, et al. Rollin'justin-design considerations and realization of a mobile platform for a humanoid upper body. In *2009 IEEE International Conference on Robotics and Automation*, pages 4131–4137. IEEE, 2009.

[51] Tommaso Furlanello, Zachary C Lipton, Michael Tschannen, Laurent Itti, and Anima Anandkumar. Born again neural networks. *arXiv preprint arXiv:1805.04770*, 2018.

[52] Chuang Gan, Jeremy Schwartz, Seth Alter, Martin Schrimpf, James Traer, Julian De Freitas, Jonas Kubilius, Abhishek Bhandwaldar, Nick

Haber, Megumi Sano, et al. Threedworld: A platform for interactive multi-modal physical simulation. *arXiv preprint arXiv:2007.04954*, 2020.

[53] James J Gibson. *The ecological approach to visual perception: classic edition*. Psychology Press, 2014.

[54] Leni K Le Goff, Oussama Yaakoubi, Alexandre Coninx, and Stephane Doncieux. Building an affordances map with interactive perception. *arXiv preprint arXiv:1903.04413*, 2019.

[55] Florian Golemo. *How to Train Your Robot - New Environments for Robotic Training and New Methods for Transferring Policies from the Simulator to the Real Robot*. Theses, Université de Bordeaux, December 2018.

[56] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.

[57] Alex Graves, Jacob Menick, and Aaron van den Oord. Associative compression networks. *arXiv preprint arXiv:1804.02476*, 2018.

[58] Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron C Courville. Improved training of wasserstein gans. In *Advances in Neural Information Processing Systems*, pages 5767–5777, 2017.

[59] Megha Gupta and Gaurav S Sukhatme. Using manipulation primitives for brick sorting in clutter. In *2012 IEEE International Conference on Robotics and Automation*, pages 3883–3889. IEEE, 2012.

[60] David Ha and Jürgen Schmidhuber. Recurrent world models facilitate policy evolution. In *Advances in Neural Information Processing Systems*, pages 2450–2462, 2018.

[61] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. *arXiv preprint arXiv:1801.01290*, 2018.

[62] Raia Hadsell, Dushyant Rao, Andrei A Rusu, and Razvan Pascanu. Embracing change: Continual learning in deep neural networks. *Trends in Cognitive Sciences*, 2020.

[63] Danijar Hafner, Timothy Lillicrap, Mohammad Norouzi, and Jimmy Ba. Mastering atari with discrete world models, 2020.

[64] William Harvey, Saeid Naderiparizi, and Frank Wood. Image completion via inference in deep generative models, 2021.

[65] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 2961–2969, 2017.

[66] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015.

[67] Matteo Hessel, Joseph Modayil, Hado Van Hasselt, Tom Schaul, Georg Ostrovski, Will Dabney, Dan Horgan, Bilal Piot, Mohammad Azar, and David Silver. Rainbow: Combining improvements in deep reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

[68] Irina Higgins, David Amos, David Pfau, Sebastien Racaniere, Loic Matthey, Danilo Rezende, and Alexander Lerchner. Towards a definition of disentangled representations. *arXiv preprint arXiv:1812.02230*, 2018.

[69] Irina Higgins, Loic Matthey, Arka Pal, Christopher Burgess, Xavier Glorot, Matthew Botvinick, Shakir Mohamed, and Alexander Lerchner. beta-vae: Learning basic visual concepts with a constrained variational framework. 2016.

[70] Irina Higgins, Arka Pal, Andrei Rusu, Loic Matthey, Christopher Burgess, Alexander Pritzel, Matthew Botvinick, Charles Blundell, and Alexander Lerchner. Darla: Improving zero-shot transfer in reinforcement learning. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 1480–1490. JMLR. org, 2017.

[71] Ashley Hill, Antonin Raffin, Maximilian Ernestus, Rene Traore, Prafulla Dhariwal, Christopher Hesse, Oleg Klimov, Alex Nichol, Matthias Plappert, Alec Radford, et al. Stable baselines, 2018.

[72] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

[73] Donald D Hoffman. The interface theory of perception. *Stevens' Handbook of Experimental Psychology and Cognitive Neuroscience*, 2:1–24, 2018.

[74] Andrew Jaegle, Vahid Mehrpour, and Nicole Rust. Visual novelty, curiosity, and intrinsic reward in machine learning and the brain. *Current opinion in neurobiology*, 58:167–174, 2019.

[75] Khurram Javed and Martha White. Meta-learning representations for continual learning. *arXiv preprint arXiv:1905.12588*, 2019.

[76] Scott P Johnson. How infants learn about the visual world. *Cognitive Science*, 34(7):1158–1184, 2010.

[77] Lukasz Kaiser, Mohammad Babaeizadeh, Piotr Milos, Blazej Osinski, Roy H Campbell, Konrad Czechowski, Dumitru Erhan, Chelsea Finn, Piotr Kozakowski, Sergey Levine, et al. Model-based reinforcement learning for atari. *arXiv preprint arXiv:1903.00374*, 2019.

[78] Tero Karras, Miika Aittala, Janne Hellsten, Samuli Laine, Jaakko Lehtinen, and Timo Aila. Training generative adversarial networks with limited data. *arXiv preprint arXiv:2006.06676*, 2020.

[79] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4401–4410, 2019.

[80] Tero Karras, Samuli Laine, Miika Aittala, Janne Hellsten, Jaakko Lehtinen, and Timo Aila. Analyzing and improving the image quality of stylegan. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8110–8119, 2020.

[81] Franz Kaufmann. Development of motion perception in early infancy. *European Journal of Pediatrics*, 154(4):S48–S53, 1995.

[82] Michał Kempka, Marek Wydmuch, Grzegorz Runc, Jakub Toczek, and Wojciech Jaśkowski. Vizdoom: A doom-based ai research platform for visual reinforcement learning. In *Computational Intelligence and Games (CIG), 2016 IEEE Conference on*, pages 1–8. IEEE, 2016.

[83] Khimya Khetarpal, Matthew Riemer, Irina Rish, and Doina Precup. Towards continual reinforcement learning: A review and perspectives, 2020.

[84] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[85] James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences*, page 201611835, 2017.

[86] Eric Kolve, Roozbeh Mottaghi, Winson Han, Eli VanderBilt, Luca Weihs, Alvaro Herrasti, Daniel Gordon, Yuke Zhu, Abhinav Gupta,

and Ali Farhadi. Ai2-thor: An interactive 3d environment for visual ai. *arXiv preprint arXiv:1712.05474*, 2017.

[87] Hema Swetha Koppula, Rudhir Gupta, and Ashutosh Saxena. Learning human activities and object affordances from rgb-d videos. *The International Journal of Robotics Research*, 32(8):951–970, 2013.

[88] Ranjay Krishna, Yuke Zhu, Oliver Groth, Justin Johnson, Kenji Hata, Joshua Kravitz, Stephanie Chen, Yannis Kalantidis, Li-Jia Li, David A Shamma, et al. Visual genome: Connecting language and vision using crowdsourced dense image annotations. *International journal of computer vision*, 123(1):32–73, 2017.

[89] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25:1097–1105, 2012.

[90] Alban Laflaquière. Unsupervised emergence of spatial structure from sensorimotor prediction. *arXiv preprint arXiv:1810.01344*, 2018.

[91] Alban Laflaquiere, Sylvain Argentieri, Olivia Breysse, Stéphane Genet, and Bruno Gas. A non-linear approach to space dimension perception by a naive agent. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3253–3259. IEEE, 2012.

[92] Alban Laflaquière, J Kevin O'Regan, Sylvain Argentieri, Bruno Gas, and Alexander V Terekhov. Learning agent's spatial configuration from sensorimotor invariants. *Robotics and Autonomous Systems*, 71:49–59, 2015.

[93] Alban Laflaquiere, Alexander V Terekhov, Bruno Gas, and J Kevin O'Regan. Learning an internal representation of the end-effector configuration space. In *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1230–1235. IEEE, 2013.

[94] Timothée Lesort. *Apprentissage continu : S'attaquer à l'oubli foudroyant des réseaux de neurones profonds grâce aux méthodes à rejeu de données*. Theses, Institut Polytechnique de Paris, June 2020.

[95] Timothée Lesort, Hugo Caselles-Dupré, Michael Garcia-Ortiz, Andrei Stoian, and David Filliat. Generative models from the perspective of continual learning. *arXiv preprint arXiv:1812.09111*, 2018.

[96] Timothée Lesort, Natalia Díaz-Rodríguez, Jean-François Goudou, and David Filliat. State representation learning for control: An overview. *Neural Networks*, 2018.

[97] Timothée Lesort, Vincenzo Lomonaco, Andrei Stoian, Davide Maltoni, David Filliat, and Natalia Díaz-Rodríguez. Continual learning for robotics: Definition, framework, learning strategies, opportunities and challenges. *Information fusion*, 58:52–68, 2020.

[98] Zhizhong Li and Derek Hoiem. Learning without forgetting. In *ECCV*, 2016.

[99] Zhizhong Li and Derek Hoiem. Learning without forgetting. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2017.

[100] Tsung-Yi Lin, Michael Maire, Serge Belongie, Lubomir Bourdev, Ross Girshick, James Hays, Pietro Perona, Deva Ramanan, C. Lawrence Zitnick, and Piotr Dollár. Microsoft coco: Common objects in context, 2015.

[101] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014.

[102] Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning face attributes in the wild. In *Proceedings of International Conference on Computer Vision (ICCV)*, December 2015.

[103] Francesco Locatello, Gabriele Abbati, Tom Rainforth, Stefan Bauer, Bernhard Schölkopf, and Olivier Bachem. On the fairness of disentangled representations. *arXiv preprint arXiv:1905.13662*, 2019.

[104] Francesco Locatello, Stefan Bauer, Mario Lucic, Sylvain Gelly, Bernhard Schölkopf, and Olivier Bachem. Challenging common assumptions in the unsupervised learning of disentangled representations. *arXiv preprint arXiv:1811.12359*, 2018.

[105] Francesco Locatello, Stefan Bauer, Mario Lucic, Gunnar Raetsch, Sylvain Gelly, Bernhard Schölkopf, and Olivier Bachem. Challenging common assumptions in the unsupervised learning of disentangled representations. In *International Conference on Machine Learning*, pages 4114–4124, 2019.

[106] Vincenzo Lomonaco, Karan Desai, Eugenio Culurciello, and Davide Maltoni. Continual reinforcement learning in 3d non-stationary environments. *CoRR*, abs/1905.10112, 2019.

[107] Jonathon Luiten, Idil Esen Zulfikar, and Bastian Leibe. Unovost: Unsupervised offline video object segmentation and tracking. In *The IEEE Winter Conference on Applications of Computer Vision*, pages 2000–2009, 2020.

[108] Max Lungarella, Giorgio Metta, Rolf Pfeifer, and Giulio Sandini. Developmental robotics: a survey. *Connection science*, 15(4):151–190, 2003.

[109] Natalia Lyubova, Serena Ivaldi, and David Filliat. From passive to interactive object learning and recognition through self-identification on a humanoid robot. *Autonomous Robots*, 40(1):33–57, 2016.

[110] Roberto Martín-Martín, Hamid Rezatofighi, Abhijeet Shenoi, Mihir Patel, JunYoung Gwak, Nathan Dass, Alan Federman, Patrick Goebel, and Silvio Savarese. Jrdb: A dataset and benchmark for visual perception for navigation in human environments. *arXiv preprint arXiv:1910.11792*, 2019.

[111] Jan Matas, Stephen James, and Andrew J Davison. Sim-to-real reinforcement learning for deformable object manipulation. *arXiv preprint arXiv:1806.07851*, 2018.

[112] Mehdi Mirza and Simon Osindero. Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784*, 2014.

[113] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

[114] Filipe Figueredo Monteiro, Andre Luiz Buarque Vieira, João Marcelo Xavier Natário Teixeira, Veronica Teichrieb, et al. Simulating real robots in virtual environments using nvidia's isaac sdk. In *Anais Estendidos do XXI Simpósio de Realidade Virtual e Aumentada*, pages 47–48. SBC, 2019.

[115] Austin Myers, Ching L Teo, Cornelia Fermüller, and Yiannis Aloimonos. Affordance detection of tool parts from geometric features. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1374–1381. IEEE, 2015.

[116] Anh Nguyen, Dimitrios Kanoulas, Darwin G Caldwell, and Nikos G Tsagarakis. Detecting object affordances with convolutional neural networks. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2765–2770. IEEE, 2016.

[117] Anh Nguyen, Dimitrios Kanoulas, Darwin G Caldwell, and Nikos G Tsagarakis. Object-based affordances detection with convolutional neural networks and dense conditional random fields. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5908–5915. IEEE, 2017.

[118] Cuong V Nguyen, Yingzhen Li, Thang D Bui, and Richard E Turner. Variational continual learning. *arXiv preprint arXiv:1710.10628*, 2017.

[119] Alex Nichol, Vicki Pfau, Christopher Hesse, Oleg Klimov, and John Schulman. Gotta learn fast: A new benchmark for generalization in rl. *arXiv preprint arXiv:1804.03720*, 2018.

[120] Sebastian Nowozin, Botond Cseke, and Ryota Tomioka. f-gan: Training generative neural samplers using variational divergence minimization. In *Advances in Neural Information Processing Systems*, pages 271–279, 2016.

[121] Seoung Wug Oh, Joon-Young Lee, Ning Xu, and Seon Joo Kim. Video object segmentation using space-time memory networks. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 9226–9235, 2019.

[122] Aaron van den Oord, Oriol Vinyals, and Koray Kavukcuoglu. Neural discrete representation learning. *arXiv preprint arXiv:1711.00937*, 2017.

[123] J Kevin O'Regan and Alva Noë. A sensorimotor account of vision and visual consciousness. *Behavioral and brain sciences*, 24(5):939–973, 2001.

[124] Pierre-Yves Oudeyer. Computational theories of curiosity-driven learning. *arXiv preprint arXiv:1802.10546*, 2018.

[125] German I Parisi, Ronald Kemker, Jose L Part, Christopher Kanan, and Stefan Wermter. Continual lifelong learning with neural networks: A review. *Neural Networks*, 2019.

[126] Emilio Parisotto, Francis Song, Jack Rae, Razvan Pascanu, Caglar Gulcehre, Siddhant Jayakumar, Max Jaderberg, Raphael Lopez Kaufman, Aidan Clark, Seb Noury, et al. Stabilizing transformers for reinforcement learning. In *International Conference on Machine Learning*, pages 7487–7498. PMLR, 2020.

[127] Niki Parmar, Ashish Vaswani, Jakob Uszkoreit, Lukasz Kaiser, Noam Shazeer, Alexander Ku, and Dustin Tran. Image transformer. In *International Conference on Machine Learning*, pages 4055–4064. PMLR, 2018.

[128] Deepak Pathak, Pulkit Agrawal, Alexei A Efros, and Trevor Darrell. Curiosity-driven exploration by self-supervised prediction. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 16–17, 2017.

[129] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[130] Federico Perazzi, Jordi Pont-Tuset, Brian McWilliams, Luc Van Gool, Markus Gross, and Alexander Sorkine-Hornung. A benchmark dataset and evaluation methodology for video object segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 724–732, 2016.

[131] David Philipona. Développement d'un cadre mathématique pour une théorie sensorimotrice de l'expérience sensorielle. 2008.

[132] David Philipona, J Kevin O'Regan, and J-P Nadal. Is there something out there? inferring space from sensorimotor dependencies. *Neural computation*, 15(9):2029–2049, 2003.

[133] David Philipona, Jk O'regan, J-P Nadal, and Olivier Coenen. Perception of the structure of the physical world using unknown multimodal sensors and effectors. In *Advances in neural information processing systems*, pages 945–952, 2004.

[134] Henri Poincaré. L'espace et la géométrie. 1895.

[135] Ameya Prabhu, Philip HS Torr, and Puneet K Dokania. Gdumb: A simple approach that questions our progress in continual learning. In *European Conference on Computer Vision*, pages 524–540. Springer, 2020.

[136] Antonin Raffin, Ashley Hill, Kalifou René Traoré, Timothée Lesort, Natalia Díaz-Rodríguez, and David Filliat. Decoupling feature extraction from policy learning: assessing benefits of state representation learning in goal based robotics. *arXiv preprint arXiv:1901.08651*, 2019.

[137] Antonin Raffin, Ashley Hill, René Traoré, Timothée Lesort, Natalia Díaz-Rodríguez, and David Filliat. S-rl toolbox: Environments, datasets and evaluation metrics for state representation learning. *arXiv preprint arXiv:1809.09369*, 2018.

[138] Aditya Ramesh, Mikhail Pavlov, Gabriel Goh, Scott Gray, Chelsea Voss, Alec Radford, Mark Chen, and Ilya Sutskever. Zero-shot text-to-image generation, 2021.

[139] Ali Razavi, Aaron van den Oord, and Oriol Vinyals. Generating diverse high-fidelity images with vq-vae-2. *arXiv preprint arXiv:1906.00446*, 2019.

[140] Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, Georg Sperl, and Christoph H Lampert. icarl: Incremental classifier and representation learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2001–2010, 2017.

[141] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 779–788, 2016.

[142] Martin Riedmiller, Roland Hafner, Thomas Lampe, Michael Neunert, Jonas Degrave, Tom Van de Wiele, Volodymyr Mnih, Nicolas Heess, and Jost Tobias Springenberg. Learning by playing-solving sparse reward tasks from scratch. *arXiv preprint arXiv:1802.10567*, 2018.

[143] Mark B Ring. Child: A first step towards continual learning. In *Learning to learn*, pages 261–292. Springer, 1998.

[144] A. A. Rusu, N. C. Rabinowitz, G. Desjardins, H. Soyer, J. Kirkpatrick, K. Kavukcuoglu, R. Pascanu, and R. Hadsell. Progressive Neural Networks. *ArXiv e-prints*, June 2016.

[145] Andrei A. Rusu, Sergio Gomez Colmenarejo, Caglar Gulcehre, Guillaume Desjardins, James Kirkpatrick, Razvan Pascanu, Volodymyr Mnih, Koray Kavukcuoglu, and Raia Hadsell. Policy distillation, 2016.

[146] Andrei A. Rusu, Matej Vecerik, Thomas Rothörl, Nicolas Heess, Razvan Pascanu, and Raia Hadsell. Sim-to-real robot learning from pixels with progressive nets. *CoRR*, abs/1610.04286, 2016.

[147] Manolis Savva, Abhishek Kadian, Oleksandr Maksymets, Yili Zhao, Erik Wijmans, Bhavana Jain, Julian Straub, Jia Liu, Vladlen Koltun, Jitendra Malik, et al. Habitat: A platform for embodied ai research. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 9339–9347, 2019.

[148] David Schiebener, Aleš Ude, and Tamim Asfour. Physical interaction for segmentation of unknown textured and non-textured rigid objects. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4959–4966. IEEE, 2014.

[149] David Schiebener, Aleš Ude, Jun Morimoto, Tamim Asfour, and Rüdiger Dillmann. Segmentation and learning of unknown objects

through physical interaction. In *2011 11th IEEE-RAS International Conference on Humanoid Robots*, pages 500–506. IEEE, 2011.

[150] Julian Schrittwieser, Ioannis Antonoglou, Thomas Hubert, Karen Simonyan, Laurent Sifre, Simon Schmitt, Arthur Guez, Edward Lockhart, Demis Hassabis, Thore Graepel, et al. Mastering atari, go, chess and shogi by planning with a learned model. *Nature*, 588(7839):604–609, 2020.

[151] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[152] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *CoRR*, abs/1707.06347, 2017.

[153] Jonathan Schwarz, Jelena Luketina, Wojciech M Czarnecki, Agnieszka Grabska-Barwinska, Yee Whye Teh, Razvan Pascanu, and Raia Hadsell. Progress & compress: A scalable framework for continual learning. *arXiv preprint arXiv:1805.06370*, 2018.

[154] Ari Seff, Alex Beatson, Daniel Suo, and Han Liu. Continual learning in generative adversarial nets. *arXiv preprint arXiv:1705.08395*, 2017.

[155] Bokui Shen*, Fei Xia*, Chengshu Li*, Roberto Martín-Martín*, Linxi Fan, Guanzhi Wang, Shyamal Buch, Claudia D'Arpino, Sanjana Srivastava, Lyne P Tchapmi, Kent Vainio, Li Fei-Fei, and Silvio Savarese. igibson, a simulation environment for interactive tasks in large realistic scenes. *arXiv preprint arXiv:2012.02924*, 2020.

[156] Hanul Shin, Jung Kwon Lee, Jaehong Kim, and Jiwon Kim. Continual learning with deep generative replay. In *Advances in Neural Information Processing Systems*, pages 2990–2999, 2017.

[157] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[158] Linda Smith and Michael Gasser. The development of embodied cognition: Six lessons from babies. *Artificial life*, 11(1-2):13–29, 2005.

[159] Stefano Soatto. Steps towards a theory of visual information: Active perception, signal-to-symbol conversion and the interplay between sensing and control. Technical report, 2011.

[160] Rupesh K Srivastava, Jonathan Masci, Sohrob Kazerounian, Faustino Gomez, and Jürgen Schmidhuber. Compete to compute. In C. J. C.

Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 2310–2318. Curran Associates, Inc., 2013.

[161] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[162] Yuval Tassa, Yotam Doron, Alistair Muldal, Tom Erez, Yazhe Li, Diego de Las Casas, David Budden, Abbas Abdolmaleki, Josh Merel, Andrew Lefrancq, et al. Deepmind control suite. *arXiv preprint arXiv:1801.00690*, 2018.

[163] Zachary Teed and Jia Deng. Raft: Recurrent all-pairs field transforms for optical flow. *arXiv preprint arXiv:2003.12039*, 2020.

[164] Yee Teh, Victor Bapst, Wojciech M Czarnecki, John Quan, James Kirkpatrick, Raia Hadsell, Nicolas Heess, and Razvan Pascanu. Distral: Robust multitask reinforcement learning. In *Advances in Neural Information Processing Systems*, pages 4496–4506, 2017.

[165] Valentin Thomas, Jules Pondard, Emmanuel Bengio, Marc Sarfati, Philippe Beaudoin, Marie-Jean Meurs, Joelle Pineau, Doina Precup, and Yoshua Bengio. Independently controllable features. *arXiv preprint arXiv:1708.01289*, 2017.

[166] Josh Tobin, Rachel Fong, Alex Ray, Jonas Schneider, Wojciech Zaremba, and Pieter Abbeel. Domain randomization for transferring deep neural networks from simulation to the real world. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 23–30. IEEE, 2017.

[167] Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, pages 5026–5033. IEEE, 2012.

[168] René Traoré, Hugo Caselles-Dupré, Timothée Lesort, Te Sun, Natalia Díaz-Rodríguez, and David Filliat. Continual reinforcement learning deployed in real-life using policy distillation and sim2real transfer. *arXiv preprint arXiv:1906.04452*, 2019.

[169] Sjoerd van Steenkiste, Francesco Locatello, Jürgen Schmidhuber, and Olivier Bachem. Are disentangled representations helpful for abstract visual reasoning? *arXiv preprint arXiv:1905.12506*, 2019.

[170] Paul Voigtlaender, Michael Krause, Aljosa Osep, Jonathon Luiten, Berin Balachandar Gnana Sekar, Andreas Geiger, and Bastian Leibe.

Mots: Multi-object tracking and segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7942–7951, 2019.

[171] Greg Wayne, Chia-Chun Hung, David Amos, Mehdi Mirza, Arun Ahuja, Agnieszka Grabska-Barwinska, Jack Rae, Piotr Mirowski, Joel Z Leibo, Adam Santoro, et al. Unsupervised predictive memory in a goal-directed agent. *arXiv preprint arXiv:1803.10760*, 2018.

[172] Nicholas Waytowich, Sean L Barton, Vernon Lawhern, Ethan Stump, and Garrett Warnell. Grounding natural language commands to starcraft ii game states for narration-guided reinforcement learning. In *Artificial Intelligence and Machine Learning for Multi-Domain Operations Applications*, volume 11006, page 110060S. International Society for Optics and Photonics, 2019.

[173] Bernard L Welch. The generalization ofstudent's' problem when several different population variances are involved. *Biometrika*, 34(1/2):28–35, 1947.

[174] Melonee Wise, Michael Ferguson, Derek King, Eric Diehr, and David Dymesich. Fetch and freight: Standard platforms for service robot applications. 2016.

[175] Chenshen Wu, Luis Herranz, Xialei Liu, Yaxing Wang, Joost van de Weijer, and Bogdan Raducanu. Memory replay gans: learning to generate images from new categories without forgetting. *arXiv preprint arXiv:1809.02058*, 2018.

[176] Yue Wu, Yinpeng Chen, Lijuan Wang, Yuancheng Ye, Zicheng Liu, Yandong Guo, Zhengyou Zhang, and Yun Fu. Incremental classifier learning with generative adversarial networks. *arXiv preprint arXiv:1802.00853*, 2018.

[177] Fei Xia, William B Shen, Chengshu Li, Priya Kasimbeg, Micael Edmond Tchapmi, Alexander Toshev, Roberto Martín-Martín, and Silvio Savarese. Interactive gibson benchmark: A benchmark for interactive navigation in cluttered environments. *IEEE Robotics and Automation Letters*, 5(2):713–720, 2020.

[178] Fanbo Xiang, Yuzhe Qin, Kaichun Mo, Yikuan Xia, Hao Zhu, Fangchen Liu, Minghua Liu, Hanxiao Jiang, Yifu Yuan, He Wang, Li Yi, Angel X. Chang, Leonidas J. Guibas, and Hao Su. Sapien: A simulated part-based interactive environment, 2020.

[179] Kai Xu, Hui Huang, Yifei Shi, Hao Li, Pinxin Long, Jianong Caichen, Wei Sun, and Baoquan Chen. Autoscanning for coupled scene re-

construction and proactive object analysis. *ACM Transactions on Graphics (TOG)*, 34(6):1–14, 2015.

[180] Ning Xu, Linjie Yang, Yuchen Fan, Jianchao Yang, Dingcheng Yue, Yuchen Liang, Brian Price, Scott Cohen, and Thomas Huang. Youtube-vos: Sequence-to-sequence video object segmentation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 585–601, 2018.

[181] Andy Zeng, Shuran Song, Kuan-Ting Yu, Elliott Donlon, Francois R Hogan, Maria Bauza, Daolin Ma, Orion Taylor, Melody Liu, Eudald Romo, et al. Robotic pick-and-place of novel objects in clutter with multi-affordance grasping and cross-domain image matching. In *2018 IEEE international conference on robotics and automation (ICRA)*, pages 3750–3757. IEEE, 2018.

Titre : Du rôle des Actions et de l'Apprentissage Automatique dans la Perception des Agents Artificiels

Mots clés : Perception, Apprentissage de Représentations, Actions, Apprentissage par Renforcement

Résumé : L'automatisation est le moyen par lequel l'espèce humaine peut se libérer du fardeau des tâches qu'elle a déjà résolues. Ces tâches sont omniprésentes dans notre vie quotidienne, à la maison ou dans un contexte professionnel. Une grande ambition dans la recherche est de créer des agents capables d'agir et de raisonner dans le monde réel, en automatisant ces tâches résolues. Pour cela, nous supposons les agents doivent construire une perception de leur environnement, tout comme les humains. La programmation directe de ces agents est impossible en raison de la complexité du monde et de ses interaction. C'est pourquoi les approches fondées sur l'apprentissage ont prévalu dans la recherche au cours des 20 dernières années. Parmi les approches d'apprentissage automatique, nous avons plusieurs sous-domaines qui abordent chacun différents aspects de la perception que les agents devraient avoir. L'apprentissage de répresentations d'états (ARE) se concentre sur l'apprentissage des représentations de l'expérience des agents. L'apprentissage continu vise à résoudre le célèbre problème d'oubli catastrophique des réseaux de neurones, qui oublient tout ce qu'ils ont appris lorsque de nouvelles données leur sont présentées. Nous avons enfin l'apprentissage par renforcement, qui vise à apprendre à résoudre une tâche en maximisant la récompense qui lui est associée, mécanisme qui est également présent chez les agents biologiques.
D'un autre côté, nous avons aussi des approches plus originales qui n'ont pas forcément les mêmes performances mais reposent sur des paradigmes prometteurs qui pourraient permettre des progrès de recherche. La robotique développementale est un sous-domaine de la robotique qui vise à développer des méthodes d'apprentissage inspirées de la biologie sur de vrais robots. Nous avons aussi ce que nous appellerons dans ce manuscrit les approches de l'agent incarné, qui sont des considérations théoriques et pratiques basées sur les théories de la perception développées en psychologie, philosophie et sciences cognitives. Dans ces théories, le rôle des actions est crucial dans le développement de la perception. Nous utiliserons cela comme base pour la plupart de nos contributions.
Dans cette thèse, nous contribuons à ces sous-domaines de recherche en développant des connaissances théoriques et des algorithmes d'application qui visent à créer des agents avec des niveaux plus profonds de perception de leur corps et de l'environnement.

Title : On the role of Actions and Machine Learning in Artificial Agent Perception

Keywords : Perception, Representation Learning, Actions, Reinforcement Learning

Abstract : Automation is the medium by which the human species can free itself from the burden of tasks it has already solved. Such tasks are omnipresent in our daily lives, at home or in a professional context. A great quest in research is to build agents that can act and reason in the real-world, automating those solved tasks. For that, agents have to build a perception of their environments, just like humans do.
Directly programming those agents is infeasible because of the complexity of the world and its interaction. That is why learning-based approaches have been prevalent in research for the past 20 years. Among the Machine Learning approaches, we have several sub-fields that each tackle different aspects of perception that agents should have. State Representation Learning (SRL) focuses on learning representations of what the agents experience. Continual Learning (CL) aims at solving the infamous catastrophic forgetting problem of neural networks, which forget everything they learned when presented with new data. We finally have Reinforcement Learning (RL), which aims at learning to solve a task by maximizing the reward associated to it, a mechanism that is also present among biological agents.
On the other hand, we also have more original approaches that do not necessarily have the same performances but are based on promising paradigms that could allow breakthroughs. Developmental robotics (Dev-Rob) is a sub-field of robotics which aims at developing biological-inspired methods for learning on real robots. We also have what we shall call in this manuscript the Embodied Agent approaches, which are theoretical and practical considerations based on theories of perception developed in psychology. In these theories, the role of actions is crucial in the development of perception. We will use this as a basis for most of our contributions.
In this thesis we contribute to those sub-fields of research by developing theoretical insights and application algorithms which aim at creating agents with deeper levels of perception of their bodies and the environment.

Institut Polytechnique de Paris
91120 Palaiseau, France