Deep reinforcement learning for dexterous hand manipulation

handed in
MASTER'S THESIS

cand. M.Sc. Jedrzej Orbik

born on the 16.07.1992
living in:
Helene-Mayer-Ring 7
80809 Munich
Tel.: 017665268490

Human-centered Assistive Robotics
Technical University of Munich

Prof. Dr.-Ing. Dongheui Lee

Supervisor:	Dr. Alejandro Agostini, M.Sc. Shile Li
Start:	03.07.2019
Intermediate Report:	16.06.2020
Delivery:	21.07.2020

TECHNISCHE UNIVERSITÄT MÜNCHEN
Human-centered Assistive Robotics
UNIV.-PROF. DR.-ING. DONGHEUI LEE

July 15, 2020

M A S T E R ' S T H E S I S
for
Jedrzej Orbik
Student ID 03689272, Degree EI

Deep reinforcement learning for dexterous hand manipulation

Problem description:

Multi-fingered hands are very flexible and capable of performing a variety of tasks on objects of diverse shapes and sizes. However, achieving a human-like dexterity for a prosthetic hand is a big challenge due to its highly complex mechanisms. To address this challenge, state-of-the-art deep reinforcement learning methods are implemented [1]. These methods require human demonstrations of high quality and complex hand-crafted reward functions to achieve the desired robustness. In this thesis, we tackle these limitations by relaxing the necessity of high quality demonstrations [2] and by automatically defining reward functions for each specific task using human demonstrations and inverse reinforcement learning [3].

Tasks:

- Literature overview of learning from demonstrations and inverse reinforcement learning.
- Implementation of learning with behaviour cloning based on hand pose estimation [4].
- Implementation of an inverse reinforcement learning approach for dexterous hand manipulation [3].
- Assessment of the stability and transferability of the approaches for different tasks and scenarios.

Bibliography:

[1] A. Rajeswaran et al. Learning Complex Dexterous Manipulation with Deep Reinforcement Learning and Demonstrations, in *ArXiv170910087* Cs, Sep. 2017.
[2] K. Lowrey, S. Kolev, J. Dao, A. Rajeswaran, and E. Todorov. Reinforcement learning for non-prehensile manipulation: Transfer from simulation to physical system, in *ArXiv180310371* Cs, Mar. 2018.
[3] J. Fu, K. Luo, and S. Levine. Learning Robust Rewards with Adversarial Inverse Reinforcement Learning, in *ArXiv1710.11248* Cs, Oct. 2017.
[4] Li, Shile, and Dongheui Lee. Point-to-Pose Voting based Hand Pose Estimation using Residual Permutation Equivariant Layer, in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2019.

Supervisor:	Dr. Alejandro Agostini, MSc. Shile Li
Start:	03.07.2019
Intermediate Report:	16.06.2020
Delivery:	21.07.2020

(D. Lee)
Univ.-Professor

Abstract

Dexterous multi-fingered robotic hands are important for embedding the robots in human-centric environment designed for human-like hands. Such general-purpose end-effectors are able to perform variety of tasks, what makes them an appealing subject for research. Unfortunately, the added dexterity does not come without cost. Dexterous hand manipulation is a challenging task, because of rich contact patterns and high-dimensional action and state spaces, which result in difficult exploration problem. Current state-of-the-art deep reinforcement learning method for dexterous hand manipulation [RKG+17] is able to deal with various dexterous hand manipulation tasks using the expert demonstrations, but its application is hindered by the necessity to meticulously handcraft the reward function.

One of the possibilities to mitigate this limitation is the use of inverse reinforcement learning (IRL) framework, which hold the promise of inference of the reward function directly from the expert demonstrations. The current methods however, has proven inefficient in the high-dimensional problems. The reason behind this is the lack of robustness of the reward function to unseen inputs and this problem is especially severe in complex, high-dimensional settings. Such inferred reward functions can be exploited during policy learning, in the very same way as the human-defined rewards, resulting in the behaviour different from the expected one.

In order to deal with this shortcoming we extend the state-of-the-art inverse reinforcement learning (IRL) [FLL17] and propose to employ the sample augmentation, reward function normalisation and masking of the state space. These extensions are evaluated with respect to state-of-the-art IRL [FLL17] and direct reinforcement learning [RKG+17] in object relocation tasks. We demonstrate that using our methods, and especially the masking of the state space, we obtain robust rewards, which provide better signal for policy learning. It allows efficient learning of complex dexterous hand manipulation tasks directly from human demonstrations without manual reward function engineering. Importantly, we are able to transfer the method to new tasks: in-hand object manipulation and tool usage without any additional task-specific engineering.

Contents

Summary of notation

Matrices are denoted with bold capital letters, vectors are bold lower case letters.

t	discrete time step
T	final time step in episode
s_t	state at time step t
a_t	action at time step t
τ	trajectory - sequence of state and actions $(s_0, a_0, s_1, ..., a_{T-1}, s_T)$
$\pi(a_t\|s_t)$	policy - probability of taking action a_t at state s_t
$r(s_t, a_t)$	immediate reward for taking action a_t at state s_t
$p(s_{t+1}\|s_t, a_t)$	transition dynamics
$V(s_t)$	state value function (value function)
$Q(s_t, a_t)$	state-action value function (Q function)
$\boldsymbol{\theta}$	policy parameters
$\boldsymbol{\psi}$	reward parameters
$\pi_{\boldsymbol{\theta}}$	policy corresponding to parameters $\boldsymbol{\theta}$
$r_{\boldsymbol{\psi}}$	reward function corresponding to parameters $\boldsymbol{\psi}$
$H(\pi_{\boldsymbol{\theta}})$	differential entropy of policy $\pi_{\boldsymbol{\theta}}$ - measure of randomness of taking different actions a_t under the policy $\pi_{\boldsymbol{\theta}}$

Chapter 1

Introduction

1.1 Motivation

The majority of the robots used today include only simple end-effectors with 1 degree of freedom. This configuration is sufficient for many repetitive industrial tasks such as painting, camera operation or welding, but is constrained to the specific task to which it was designed [MLS17].
On the other hand the manipulation with dexterous end-effector similar to human hand relaxes this constraint and allows the application of robot in the variety of complex tasks. This flexibility of dexterous hand is especially important if we aim to blend the robots into human-centric environment created with ergonomics in mind.
Besides, dexterous hand manipulators matter in endeavours to create the robots resembling humans. Such humanoid robots may play a social role next to their human partners as assistive robots e.g. as a teaching aid or in hospitals for rehabilitation services and healthcare. In such scenarios their similarity to humans would facilitate engaging interaction, which is important for education or caregiving [BS99]. These interactions require robot coherency also with respect to end-effectors. They should match the user expectation with respect to appearance and their capabilities. It makes dexterous hand manipulators indispensable for further advancement in human-robot interaction.

1.1.1 Dexterous manipulation

Manipulation is one of the main research areas of robotics. It depends on robotic perception, which should provide sufficient sensory information for the task execution, and the robot's mobility if desired [DBP+16]. The end-effectors with many degrees of freedom which are able to perform the variety of different grasps can be classified as dexterous manipulators.
Human hands are example of such versatile manipulator. They allow the execution of daily life tasks, such as object relocation, tool usage, writing, painting. It makes

the human-like robotic hand a natural choice when looking for and example for a versatile dexterous manipulator.

Dexterous manipulation does not come without a cost. It is a very demanding task, that limits its robotics use. The current state-of-the-art (SoA) learning algorithms, such as deep reinforcement learning (DRL), which holds promise of autonomous learning with minimal human supervision, still struggle in the environments of high dimensionality of state and action spaces and require many samples which may not be feasible to collect from the real hardware. Such examples can be collected from the simulated robotic hand, but transfer of such learned policies to real world require a substantial amount of additional engineering work [OAB+18].

This master's thesis focuses on learning dexterous manipulation in the simulator without consideration of the later transfer of learned policies to hardware.

One of the approaches to significantly lower the required amount of samples in various learning algorithms are the expert demonstrations [KP09, NMA+18, HVP+17]. They provide a valuable prior for learning the required task even in complex environments [RKG+17].

Commonly used methods such as kinaesthetic teaching cannot be used for complex manipulators, such as the ones used for dexterous manipulation. Suitable demonstrations can be obtained either with professional motion capture systems designed to this end [KT15] or in simpler, imperfect setups with the hand pose estimation (HPE) algorithms coped with motion targeting method [LL18, AGK18]. Such demonstrations can be directly performed in the simulator or on a real hardware to acquire the required expert samples.

The overview of our framework for learning from demonstrations is presented in Figure 1.1. We use the single depth camera with a deep learning model [LL18] to estimate the human hand pose. Estimations are used as the input to inverse kinematics to retarget the motion in the simulation environment. These trajectory are adjusted on-line with particle swarm optimisation (PSO) in a task-specific manner to facilitate the task execution and capture successful demonstrations. Then the samples of the demonstrations are provided for the inverse reinforcement learning to learn a policy $\pi_{\boldsymbol{\theta}}$. In the end, the policy is used for the control in unseen task configurations.

1.1.2 Task statement

In this master's thesis we want to achieve the capabilities to relocate the objects with the dexterous robotic hand without necessity to specify reward function. The learning process is going to be performed using OpenAI Gym environments [BCP+16] with MuJoCo physics engine [TET12].

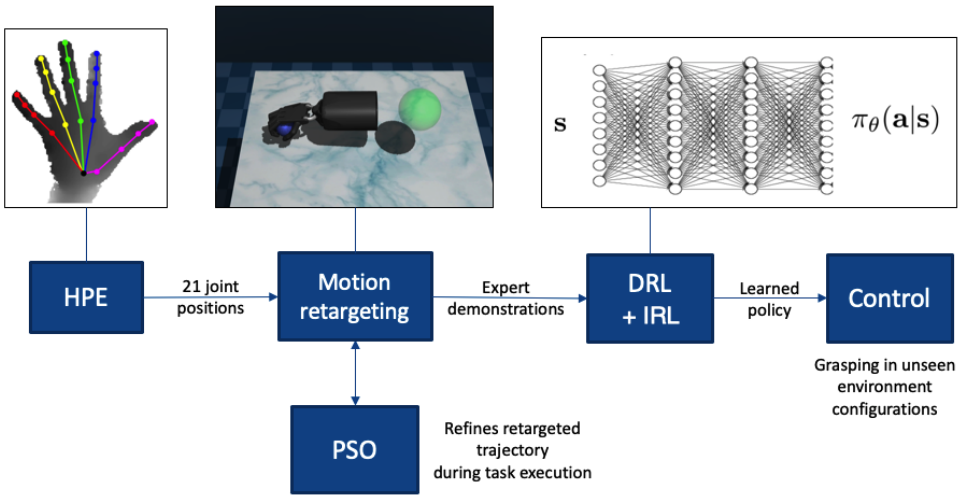

Figure 1.1: Overview of the framework. The inferred hand pose with hand pose estimation algorithm (HPE) is used to perform retargeted motion in the simulator. It is done with the aid of particle swarm optimisation (PSO), which improves the quality of the demonstrations. The recorded demonstrations are used as a prior for learning with deep reinforcement learning approach (DRL) and inverse reinforcement learning (IRL).

1.2 Related Work

Reinforcement learning (RL) application for dexterous manipulation has been a subject to a variety of work. Demo Augmented Policy Gradients (DAPG) approach by Rajeswaran et al. [RKG+17] can be seen as the state of the art for learning dexterous hand manipulation and is used in this work as a baseline. It is a policy gradient, model-free deep reinforcement learning approach, which uses the human demonstrations to cope with the exploration difficulties in high-dimensional state and action space. Learning is performed in the MuJoCo simulator [TET12] without transfer to real hardware.

In the paper by Open AI [OAB+18] the same simulator is used for learning the object manipulation with Proximal Policy Optimization (PPO) algorithm and extensive work has been devoted to allow zero-shot transfer to real world using the domain randomisation. This approach has been extended in their later work [OAA+19] where the automatic domain randomisation has been introduced.

Earlier work by Popov et al. [PHL+] has introduced an extension to Deep Deterministic Policy Gradient algorithm to learn the dexterous manipulation from multiple simple dexterous hands performing the task in parallel and thus improving the learning speed.

Other works have focused on learning directly on the robot. Kumar et al. [KGTL16]

trained set of policies in data-driven way and uses nearest neighbour method for the policy selection during evaluation. The work by Falco et al. [FASL18] uses the combination of reactive control, based on the tactile sensor and RL with vision input to learn dexterous hand manipulation.

Learning from demonstrations for soft dexterous robotic hand has been achieved by Gupta et al. [GELA17] by the selection of the most suitable demonstration for the guided policy search based learning.

A model-based RL approach for learning on the robot has been presented in work by Nagabandi et al. [NKLK19]. It is an off-policy method, which uses an ensemble of models for model fitting. They have trained a model which could be used for planning of variety of tasks, different from the task it was originally trained on - model could be used to learn the control for writing with pencil or rotating the valve after training on manipulate of Baoding balls in hand.

Learning directly from complex low-cost end-effector [AZH+19] and Allegro robotic hand has been achieved with a model of low complexity as proposed by Rajeswaran et al. [RLTK17]. In their work, diverse representations of policy were evaluated in the task of valve manipulation. After learning simple linear policy and what was called radial basis function policy the actor was able to achieve state-of-the-art learning performance with considerably simpler model than neural network function approximator. It allowed learning of the policy under 10 hours directly in real world and avoided the difficult transfer of the learned policy from a simulator.

State-of-the-art inverse reinforcement learning (IRL) in generative adversarial networks (GAN) setting [GPM+] has been proposed by Finn et al. [FCAL16] in trajectory-centric approach. An extension to their work was adversarial inverse reinforcement learning (AIRL) by Fu et al. [FLL17], a method with specific form of discriminator, which disentangles the reward function from the task dynamics. Ho et al. [HE16] have suggested to directly use binary classifier as the discriminator for the imitation learning algorithm. This form does not have explicit reward function representation, so it does not give any insight into the task, but the approach can be used to learn the policy from the demonstrations.

While the previous state-of-the-art work on Deep Reinforcement Learning (DRL) application for dexterous hand manipulation [RKG+17] allows the efficient learning, the application of the method is limited by the necessity to engineer the reward function for each of the tasks separately. Moreover, the incorrect reward function specification may easily lead to the unexpected behaviour [CA16]. In extreme cases the manipulation of the environment by the actor may even result in highly undesirable reward tampering [EH19]. This danger of reward misspecification is avoided when using IRL in GAN setup, because at convergence the policy matches the features from the expert demonstration, what constrains the possible actions taken in the environment [AN04].

1.3 Contribution

The work by Rajeswaran et. al [RKG+17] is a state-of-the-art method for learning dexterous hand manipulation. In this original work the reinforcement learning learns a policy for a specific task from the expert demonstrations and provided reward function. Our method is an extension for inverse reinforcement learning (IRL) to cope efficiently with the high-dimensional tasks of dexterous hand manipulation.

Transferability - we overcome the limitation that baseline method, which requires the specification of the reward function for each task. Our method is able to learn challenging dexterous hand manipulation solely from the demonstrations.

Learning efficiency - proposed novel inverse reinforcement learning methods for dexterous hand manipulation achieve high success rate with fewer samples than current state-of-the-art inverse reinforcement learning.

Evaluation of robustness of the reward function - the quality of the reward function is evaluated qualitatively and quantitatively to indicate the difference between our method and the vanilla inverse reinforcement learning.

Our method is compared to the baseline performance and SoA [FLL17] in the following setups:

- Learning the object relocation, the main task considered in this work, from 25 expert demonstrations provided by Rajeswaran et al. [RKG+17]
- Transfer of the method to new tasks to assess the versatility of the learning method in various dexterous hand manipulation tasks without reward function engineering

We are able to efficiently learn in multiple dexterous hand manipulation tasks without manual reward function specification. Our method outperforms the state-of-the-art AIRL method [FLL17] in terms of achieved success rate of policy in the new tasks and produces more robust reward functions.

Chapter 2

Basic elements

2.1 Reinforcement Learning

Reinforcement learning is the problem formulation for the sequential decision-making. It is often formulated as a Markov Decision Process, where the actor is expected to learn from the interactions with the environment. At each step of the interaction the agent receives the reward for the action a_t taken in the state s_t (Figure 2.1).

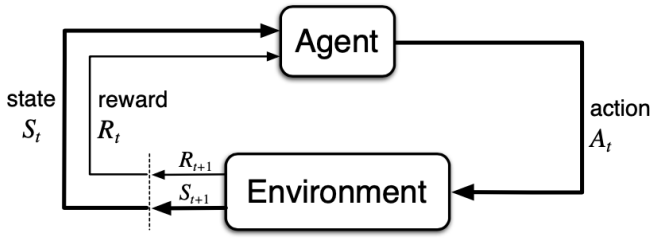

Figure 2.1: Agent interacts with the environment in the Markov Decision Process [SB18].

Over time, a well-specified reinforcement learning algorithm explores the environment during training by visiting the environment states, taking different actions and improving its policy by learning to take actions that lead to maximisation of returns in the long term.

The Markov Decision Process (MDP) problem to be solved by reinforcement learning is defined as [SB18]:

- S - set of states s_t
- $A(s_t)$ - set of possible actions a_t in state s_t
- R - set of all possible rewards $\in \mathbb{R}$
- $p(s_{t+1}|s_t, a_t)$ - transition dynamics

- $\pi(s_t, a_t)$ - policy, decision-making rule

where it is customary to assume, that the reward function $r(s_t, a_t) \in R$ is given.

We define the optimisation problem as finding the optimal policy π^* which maximises the expectation of the reward r over all time steps t:

$$\max \sum_{t=1}^{T} \mathbb{E}_{\tau \sim p_{\boldsymbol{\theta}}(\tau)}[r(s_t, a_t)] \tag{2.1}$$

under the probability distribution of the sequences:

$$p_{\boldsymbol{\theta}}(s_1, a_1, ..., s_T, a_T) = p(s_1) \prod_{t=1}^{T} \pi_{\boldsymbol{\theta}}(a_t|s_t) p(s_{t+1}|s_t, a_t) \tag{2.2}$$

where we define sequence τ as trajectory $s_1, a_1, ..., s_T, a_T$.
The policy defines the control by means of the learned parameters $\boldsymbol{\theta}$ and is represented as the conditional probability distribution $\pi_\theta(a_t|s_t)$ of actions a given state s.
The objective is to find the optimal parameters $\boldsymbol{\theta}$ which will maximise the sum over expectations of the rewards:

$$\boldsymbol{\theta}^* = \arg\max_{\boldsymbol{\theta}} \sum_{t=1}^{T} \mathbb{E}_{\tau \sim p_{\boldsymbol{\theta}}(\tau)}[r(s_t, a_t)] \tag{2.3}$$

2.2 Dexterous hand manipulation task

This master's thesis considers the task of the object relocation using the robotic dexterous hand depicted in Figure 2.2. In each episode the initial position of the object and the target are randomised.

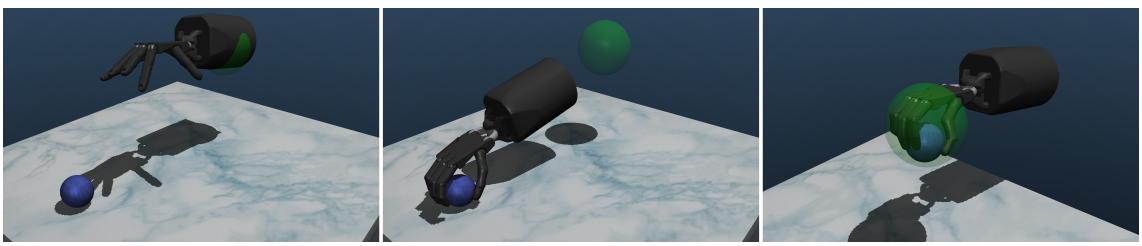

Figure 2.2: The goal of the task is to relocate the blue ball from the initial position to the target position marked as the green sphere.

2.2.1 State space

Since the learning is performed in the simulation environment there is unrestricted access to the information about the current state of the task execution and thus we are working in fully-observed environment. This allows the use of the Markov property, because we can correctly assume that the current state is independent of the previous state of the environment. Taking into consideration previous state cannot give us more information than we get from the current state. This makes the samples i.i.d. what is an important assumption made in many reinforcement learning or machine learning algorithms.
We are working in the episodic task setting, since each execution is limited by the given maximal number of steps. Each execution finishes either in success or failure of the task.
At each time step the complete information about the simulated environment is available: joint position, forces in joints, readings from artificial touch sensors etc.
In the original work by Rajeswaran et al. [RKG+17] the observation consists of:

- position and orientation of hand base (6 dimensions)
- hand joint angles (24 dimensions)
- $\overrightarrow{p_O p_H}$ (3 dimensions)
- $\overrightarrow{p_T p_O}$ (3 dimensions)
- $\overrightarrow{p_T p_H}$ (3 dimensions)

giving total state space dimension of 39, where:

- p_H is position below the middle of hand's palm where, if the object is placed, we expect the grasping to be secure
- p_O is position of object
- p_T is position of target location

all positions are given in world cartesian coordinate system.
While this does not contradict with the previous findings that the use of neural network function approximators allows the use of raw observations without feature engineering in end-to-end manner [MKS+15, HGEJ17] this result shows that proper feature engineering will significantly reduce sample complexity for this policy gradient method.

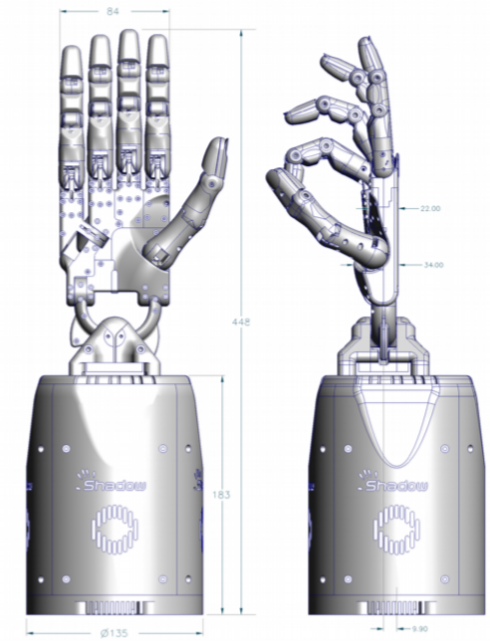

Figure 2.3: Depiction of the used model of Shadow Robot Dexterous Hand with defined dimensions [Sha].

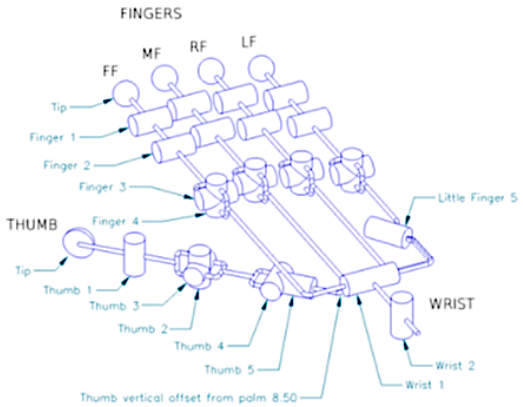

Figure 2.4: Kinematics of the robotic hand model. This figure presents 24 rotational joints of the hand [Sha].

2.2.2 Action space

The action space has dimension of 30: 24 dimensions belongs to the hand's rotational joints, 3 to the rotation and 3 to translation of the base robotic hand as depicted in Figure 2.3, 2.4. The input can be either the position, velocity or torque/force control.
In robotics however it is a common practice to use the position control for the dexterous hand manipulation [OAB+18, OAA+19]. Based on these previous works and the results of performed evaluation with particle swarm optimisation in Section 3.3 it became apparent that it is the optimal control input.

2.3 Expert demonstrations

The demonstrations are carried out in the setup illustrated in Figure 2.5. We execute the task of grasping the ball in simulation environment and relocate it to the given target position, as described in Section 2.2. When performing the demonstration the readings from a single depth camera are fed to the deep learning regression model to infer the hand joints positions in the cartesian coordinate system. The direct interpretation of the target joints angles is not possible due to kinematics discrepancies of the human and robotic hand. So firstly, we perform the proper scaling of the inferred Cartesian joints positions. Then the target position of the

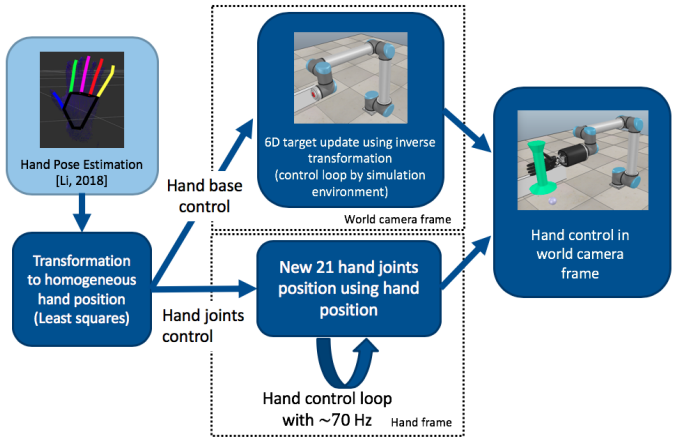

Figure 2.5: The scheme of the our demonstration acquisition architecture. The readings from the camera are fed to the deep neural network for pose estimation and the outputs are introduced to the closed loop inverse kinematics. Visualisation of grasping presented here is in V-REP simulation environment [RSF13] used in previous work.

hand base is determined by the least-squares estimation of transformation parameters between point patterns of the knuckles and the wrist of the human and robotic hand. These positions are used to find the target joint angles using the implemented inverse kinematics. The result is an input, which are fed directly to the controller in compliance with the action space described in Section 2.2.2.
The demonstrations are imperfect because of the lack of the haptic feedback during the execution of the task, errors in hand pose estimation from the hand self-occlusions and delay of the system. Because of lack of haptic feedback during the demonstrations the hand is closed on the object without guarantee the object is firmly grasped, what implies the low success rate of the demonstrations. This does not allow us to make the assumption that the policy of the expert is optimal. The solution of the IRL with the notion of suboptimality proposed in the work by Ziebart et al. [ZBD08] introduced in Section 3.2.2 has to be followed.

Chapter 3

Methodology

In this chapter we describe the theoretical tools from state of the art, which are employed in the contributions presented in the following Chapter 4. Here we present deep reinforcement learning, inverse reinforcement learning together with its shortcomings and particle swarm optimisation used for the acquisition of the successful demonstrations of dexterous hand manipulation.

3.1 Deep Reinforcement Learning

Deep reinforcement learning is a fairly new machine learning method which has proven to be successful in the variety of different tasks [MKS+13, MKS+15, SHM+16, VBC+19b]. It extends the reinforcement learning introduced in Section 2.1 using artificial deep neural networks to learn from rich inputs. Specifically, the neural network models approximate the elements of the reinforcement learning: it approximates the dynamics of the environments $p(s_{t+1}|s_t, a_t)$ in case of model based RL or the actor policy $\pi(a_t|s_t)$ for model-free RL.

Deep reinforcement learning is able to learn from any input any required output if we apply big enough model. It has been shown in previous contributions when mastering different challenges: board game go [SSS+17], Real-Time Strategy Starcraft 2 [VBC+19a], solving Rubik's cube with a robot hand [OAA+19]. All these algorithms has been solved using one general algorithm.

The adaptability of this algorithm resembles the flexibility of the human's brain and thus can be seen as a sole solution to creating general artificial intelligence following the example of the brain. In the different works it has been shown, that the auditory cortex is able to learn the visual functions [RPKS92] or the vision is possible with the nerves from tongue [NPAF15] similarly as the RL algorithms are able to learn different functions from diverse inputs. This flexibility of reinforcement learning suggests that it may be the general solution for many challenging decision-making problems, including robotics.

3.2 Inverse reinforcement learning

Reinforcement learning is a powerful general framework for learning control from experience, but its applicability is often hindered by the requirement of the onerous engineering of the reward function and features [CLB+17].
As described in Section 2.1 the reward function $r(s_t, a_t)$ is usually assumed to be given. In reality the reward has to be handcrafted to achieve the desired behaviour of the learned policy π. This presents various difficulties and pitfalls.
Reward function specification may be impractical in some setups. Consider the specification of reinforcement learning reward for autonomous driving car or for carrying out a dialogue with a human. There is no viable way of defining it in a concise form.
The misspecified reward function may lead to undesired behaviours, so that even though the learned behaviour will be optimal with respect to the returned rewards, it will not comply with the user's expectations leading to actual AI safety problems [CA16, AOS+16]. In some cases the manipulation of the environment by the actor may lead to highly undesirable reward tampering [EH19].

Inverse reinforcement learning [AN04] (also called Inverse Optimal Control or Apprenticeship Learning) holds the promise of inferring the reward function $r(s_t, a_t)$ from the expert demonstrations given as the samples of the optimal or near optimal policy π_E (expert policy). This aims to avoid the manual engineering of the reward function by inferring it from the provided samples of the expert policy - demonstrations.
Inverse reinforcement learning can be seen as a form of imitation learning, but by definition IRL lends us a way to find a reward function based on the samples from the expert. As the result the agent may outperform the expert when using the learned reward function [BGN19] or the reward function itself may grant information about the characteristics of the task, what may be of interest to us. In this way this approach is more appealing than the standard behaviour cloning method [SHKM92, Pom89].

The early research of apprenticeship learning has lead to the specification of the objective in a form, which was aiming to maximise the feature expectations with respect to the expert demonstrations consisting of the sampled states s_t from the expert trajectories [AN04]:

$$\mu(\pi) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t \phi(s_t) | \pi\right] \in \mathbb{R}^k \tag{3.1}$$

where ϕ defines the features of state s_t. By solving the maximum likelihood problem $\|\mu(\tilde{\pi_{\boldsymbol{\theta}}}) - \pi_E\}\|_2$ we find the policy $\pi_{\boldsymbol{\theta}}$ which is close to the expert's policy on the unknown reward function $r(s_t, a_t)$.

This method allows finding the policy $\pi_{\boldsymbol{\theta}}$ for which the feature expectations will match those from the expert trajectories, but there are many possible policies, which satisfy the optimisation objective matching the feature expectations. This problem has been addressed in the inverse reinforcement learning framework presented in Section 3.2.2 and used in the following state-of-the-art works.

3.2.1 State-of-the-art inverse reinforcement learning methods

State-of-the-art inverse reinforcement learning is implemented in the form of generative adversarial networks [GPM+] as depicted in Figure 5.3. It consists of two models: generator G and discriminator D competing against each other. The discriminator's aim is to distinguish between the samples from the target distribution (positives) and the samples from generator (negatives). The generator tries to maximise the probability that the discriminator will make a mistake.

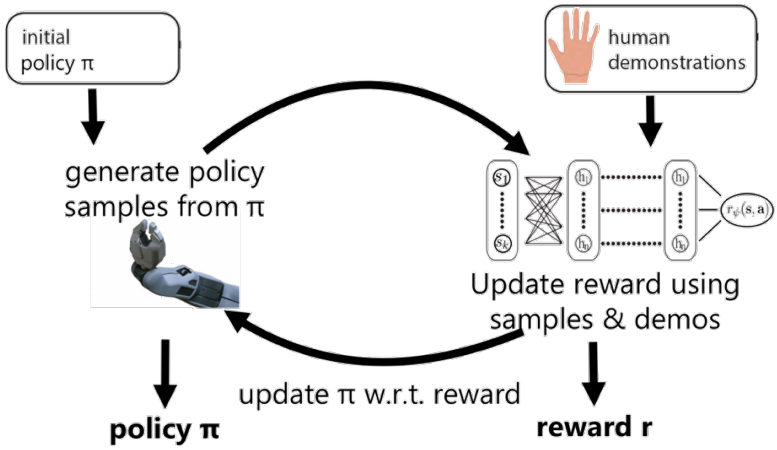

Figure 3.1: The inverse reinforcement learning general scheme [Fin].

Current state-of-the-art reinforcement learning methods learning from demonstrations:

GCL - Guided Cost Learning [FLA16] is the formulation of the inverse reinforcement learning, which contrary to two next methods works on trajectory level ($r(\tau)$) instead of state-action pairs ($r(s, a)$). The reward function is found as the optimiser with respect to the loss function:

$$\mathcal{L}_{\mathrm{GCL}}(\boldsymbol{\psi}) = \frac{1}{N} \sum_{\tau_i \in \mathcal{D}_{\mathrm{demo}}} c_{\boldsymbol{\psi}}(\tau_i) + \log \frac{1}{M} \sum_{\tau_j \in \mathcal{D}_{\mathrm{samp}}} z_j \exp(-c_{\boldsymbol{\psi}}(\tau_j)), \tag{3.2}$$

where $c_{\boldsymbol{\psi}}(\tau) = -r_{\boldsymbol{\psi}}(\tau)$ and z is the importance weight. It leads to alternating updates of the policy and the cost function, a form which resembles generative ad-

versarial networks. This perspective is presented on Figure 3.1. The calculation of importance weights z is necessary, because the samples from distribution $\mathcal{D}_{\text{samp}}$ ought to be the samples induced by the parameters $\boldsymbol{\psi}$, what would require training of the policy until convergence, what is unfeasible. Because we are sampling from a different distribution the importance sampling and the weights z have been incorporated.

GAIL - Generative Adversarial Imitation Learning [HE16] is not a IRL method, because it does not have explicit representation of the reward function to train the policy. Instead, it uses binary classifier as in typical GAN setting and seeks to find the saddle point of the expression:

$$\mathbb{E}_{\pi_{\boldsymbol{\theta}}}[\log(D_{\boldsymbol{\psi}}(s,a))] + \mathbb{E}_{\pi_E}[\log(1 - D_{\boldsymbol{\psi}}(s,a)] - \lambda H(\pi_{\boldsymbol{\theta}}), \tag{3.3}$$

which is the optimal cross-entropy loss for classification of the samples between the π_E - expert policy and $\pi_{\boldsymbol{\theta}}$ - sampling policy. This term is regularised by the entropy term $H(\pi_{\boldsymbol{\theta}})$, following the max-ent reinforcement learning formulation [ZBD08]. $D_{\boldsymbol{\psi}}(s,a)$ is the output of the discriminator given state and action pair, is the inferred probability that sample is coming from the expert policy distribution. Training results in the policy which in optimum matches the expert without specifying the reward function in the process explicitly.
This method achieves similar goal as the IRL - learning without the specification of the reward function. It is one of the major, state-of-the-art method for imitation learning with RL and hence it was reasonable to compare its performance to inverse reinforcement learning approaches for the reference.

AIRL - Adversarial Inverse Reinforcement Learning [FLL17] aims to find a reward function, which will be robust to the changes of the environment by disentangling the reward from the transition dynamics. It is an inverse reinforcement learning method work in the framework of GAN, similarly as GCL [FLA16], but in sample-centric notion. In this method has been shown, that the effect of the environment dynamics can be minimised formulating the discriminator D as:

$$D_{\boldsymbol{\psi},\boldsymbol{\phi}}(s_t, a_t, s_{t+1}) = \frac{\exp(f_{\boldsymbol{\psi},\boldsymbol{\phi}}(s_t, a_t, s_{t+1}))}{\exp(f_{\boldsymbol{\psi},\boldsymbol{\phi}}(s_t, a_t, s_{t+1})) + \pi_{\boldsymbol{\theta}}(a|s)}, \tag{3.4}$$

where learned function $f_{\boldsymbol{\psi},\boldsymbol{\phi}}(s_t, a_t, s_{t+1})$ is restricted to reward approximator $g_{\boldsymbol{\psi}}$ and a shaping term $h_{\boldsymbol{\phi}}$:

$$f_{\boldsymbol{\psi},\boldsymbol{\phi}}(s_t, a_t, s_{t+1}) = g_{\boldsymbol{\psi}}(s_t, a_t) + \gamma h_{\boldsymbol{\phi}}(s_{t+1}) - h_{\boldsymbol{\phi}}(s_t) \tag{3.5}$$

and term $h_{\boldsymbol{\phi}}$ happens to be the value function $V(s_t)$. The justification for this result is provided in the reference [FLL17].
Instead of using state and action for the input to the reward function it is possible to parametrise $g_{\boldsymbol{\psi}}(s_t)$ only as a function of state. It allows the disentanglement

from the environment dynamics and vastly lowers the input space making it more applicable also in practice.
All three methods have been evaluated in Section 5.2.

While these IRL methods attempt to reason the reward function directly from the demonstrations, they are subject to the same exploitation as the human-defined reward functions. The more careful work has to be done to mitigate this problem.

3.2.2 Max-ent formulation of inverse reinforcement learning

The maximum entropy (max-ent) formulation of inverse reinforcement learning resolves the ambiguity of the reward function specification given the expert demonstrations.
In the framework proposed by Ziebart et al. [ZBD08] the probability $p(\tau|\boldsymbol{\psi})$, which allows the notion of suboptimality of the expert, defined as:

$$p(\tau|\boldsymbol{\psi}) = \exp(r_{\boldsymbol{\psi}}(\tau))\frac{1}{Z(\boldsymbol{\psi})} \propto \exp(r_{\boldsymbol{\psi}}(\tau)), \tag{3.6}$$

with $Z(\boldsymbol{\psi})$ as the normalising partition function, is the probability of taking the trajectory τ under the linear reward function:

$$r_{\boldsymbol{\psi}}(s_t) = \boldsymbol{\psi}^T\phi(s_t),\ r_{\boldsymbol{\psi}}(\tau) = \sum_t \boldsymbol{\psi}^T\phi(s_t), \tag{3.7}$$

where ϕ defines the features of state s_t.
Similarly as before, we solve the maximum likelihood learning problem by maximising the probability of trajectories τ_E taken by the demonstrator:

$$\max_{\boldsymbol{\psi}} \frac{1}{N}\sum_1^N \log p(\tau_E|\boldsymbol{\psi}) = \max_{\boldsymbol{\psi}} \frac{1}{N}\sum_1^N r_{\boldsymbol{\psi}}(\tau) - \log Z(\boldsymbol{\psi}) \tag{3.8}$$

and taking as the partition function:

$$Z(\boldsymbol{\psi}) = \int p(\tau)\exp(r_{\boldsymbol{\psi}}(\tau)), \tag{3.9}$$

we arrive at the gradient of loss $\mathcal{L}$:

$$\nabla_{\boldsymbol{\psi}}\mathcal{L} = \frac{1}{N}\sum_{i=1}^N \nabla_{\boldsymbol{\psi}} r_{\boldsymbol{\psi}}(\tau_{E,i}) - \frac{1}{Z(\boldsymbol{\psi})}\int p(\tau)\exp(r_{\boldsymbol{\psi}}(\tau))\nabla_{\boldsymbol{\psi}} r_{\boldsymbol{\psi}}(\tau)d\tau. \tag{3.10}$$

The application of the formula of the expectation value:

$$\nabla_{\boldsymbol{\psi}}\mathcal{L} = \mathbb{E}_{\tau_E\sim\pi_E(\tau)}[\nabla_{\boldsymbol{\psi}} r_{\boldsymbol{\psi}}(\tau_E)] - \mathbb{E}_{\tau\sim p(\tau|\boldsymbol{\psi})}[\nabla_{\boldsymbol{\psi}} r_{\boldsymbol{\psi}}(\tau)], \tag{3.11}$$

leads to a gradient based algorithm, which matches the features of the sample estimate of the expert policy π_E and deals with the reward function ambiguity problem. It can be shown, that this is equivalent to the reinforcement learning objective function [Zie10]:

$$\sum_t \mathbb{E}_{\pi_{\boldsymbol{\theta}}}[r(s_t, a_t)] + \omega\, \mathbb{E}_{\pi_{\boldsymbol{\theta}}}[H(\pi_{\boldsymbol{\theta}}(a_t|s_t))], \tag{3.12}$$

where H is the differential entropy of the policy $\pi_{\boldsymbol{\theta}}$ defined as:

$$H(\pi_{\boldsymbol{\theta}}(a_t|s_t)) = -\,\mathbb{E}_{\pi_{\boldsymbol{\theta}}}[\log \pi_{\boldsymbol{\theta}}(a_t|s_t)] \tag{3.13}$$

and ω is the entropy regularisation factor ≥ 0.
The objective is then the maximisation of the reward under the constraint to keep the policy as random as possible.

3.3 Optimisation of demonstrations with PSO

According to the general scheme in Figure 1.1 particle swarm optimisation is used to assist motion retargeting during demonstration acquisition, to successfully accomplish the task. PSO is a task-specific optimisation dependent on the specified utility function. During object relocation we want to move the fingers towards the object increasing the quality of the grasp. Ideally, this prevents slipping during relocation and raises the success rate of task execution by human demonstrator.
Particle swarm optimisation is a type of stochastic optimisation method. Randomly initialised swarm population consists of particles, which are candidates for the solution of optimisation problem. The update of the particles trying to optimise the utility (energy function) is governed by the following formula:

$$v_{i,t} = v_{i,t-1} + c_1 * (p^{\text{best}}_{i,t-1} - p_{i,t-1}) + c_2 * (g^{\text{best}}_{t-1} - p_{i,t-1}), \tag{3.14}$$

$$p_{i,t} = p_{i,t-1} * v_{i,t}, \tag{3.15}$$

where $v_{i,t}$ is the velocity of particle with index i at the iteration step t out of defined maximum N steps. p is particle position, g^{best}_{t-1} is the best global particle position (with currently the lowest energy function), $p^{\text{best}}_{i,t-1}$ and $p_{i,t-1}$ are the current particle position and it's current best position. c_1 and c_2 are the system parameters, which define the exploration potential of the swarm and individual particles and number of particles M can be adjusted as a trade-off between optimisation quality and processing speed. The distribution of initial positions of the particles is dependent on the output from inverse kinematics.
The evaluation of utility for each particle is performed in simulation in order to take into account the external forces acting on the agent and possible non-linearities of the environment. For the object relocation MDP we adopt the energy function proposed by Antotsiou et al. [AGK18]:

$$E(x, y, object) = \omega_{pose} E_{pose}(x, y) + \omega_{task} E_{task}(y, object) \tag{3.16}$$

The optimisation of the demonstrations can be used both in the offline and online manner, but because the fitness of each of the particles is evaluated in the simulator the computational cost is considerable. The frequency of retargeting algorithm control frequency decreases from ∼200 Hz down to ∼5 Hz if run on the multicore Intel i5 CPU. In the online optimisation it makes the simulation significantly less responsive, what was unacceptable for some subjects during the experiments. To alleviate it we activate the computationally expensive PSO only if the distance between the hand and object is smaller than defined threshold of 0.075 m.
Importantly, after the hyperparameter tuning PSO delivers the needed aid to successfully perform the difficult relocation task using a single tracking camera and HPE algorithm.
The scheme of Particle Swarm Optimisation is presented in Figure 3.2. The optimisation is performed for specified N number of iterations or until convergence, when the fitness function does not change above the defined small threshold. The process is repeated for T steps - until the final step of the episode. The list of all PSO hyperaparameters has been included in Appendix A.

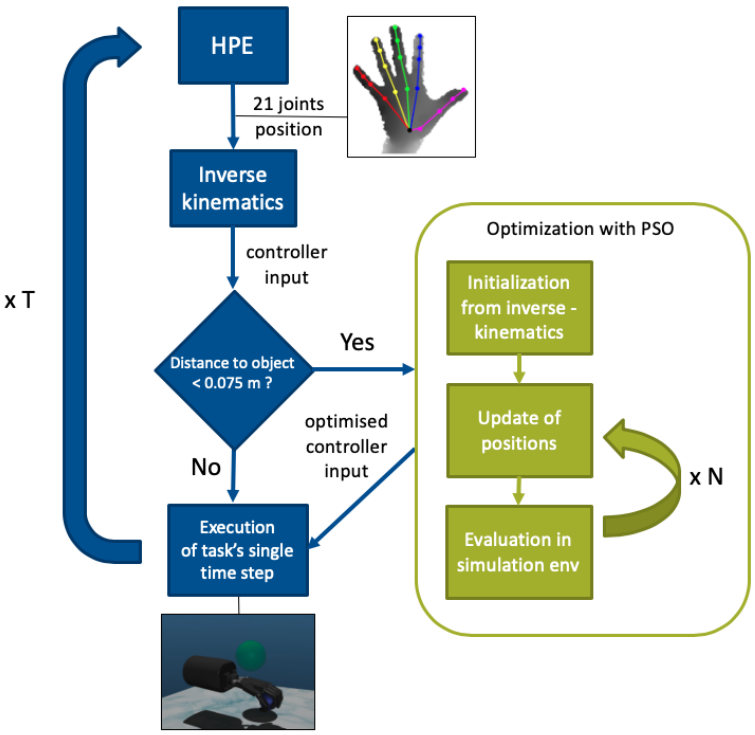

Figure 3.2: The scheme of the proposed Particle Swarm Optimisation application for our demonstrations acquisition setup.

Chapter 4

Dexterous hand manipulation without reward engineering

We aim for the similar learning capacity with inferred reward function as the state-of-the-art baseline method [RKG+17]. The avoidance of manual specification of the reward function in the high dimensional state space can be considered a substantial advancement of the state-of-the-art allowing wider application of reinforcement learning to robotics.
The initial positive results of learning with AIRL depicted in Figure 5.3 could not be confirmed when aiming to use the learned reward for learning from scratch. During learning the policy exploits the deficiencies of the learned reward function by providing inputs, which produce high rewards without achieving the goal of the task.
In order to address the problem of low sample efficiency of IRL for dexterous hand manipulation and low robustness of the reward function, which prevents efficient learning we propose novel methods, which improve the efficiency of state-of-the-art inverse reinforcement learning in dexterous hand manipulation.

4.1 Reward normalisation

Targets normalisation has been studied in the past for Temporal Difference methods by van Hasselt et al. [vGG+16]. Target normalisation has lead to simplification of the hyperparameters search and learning stability. Normalisation of the reward especially important when using the augmented policy gradient with non-stationary reward function. According to DAPG algorithm the policy gradient is augmented by gradient of demonstration samples according to the formula [RKG+17]:

$$g_{\text{aug}} = \sum_{(s,a)\in p_\pi} \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(a|s) A^\pi(s,a) + \sum_{(s,a)\in p_D} \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(a|s) w(s,a), \tag{4.1}$$

with $p_{\pi_{\boldsymbol{\theta}}}$ being the probability distribution induced by the policy $\pi_{\boldsymbol{\theta}}$, p_D the prob-

ability distribution of the given demonstrations and $w(s,a)$ the weighting factor. The advantage function $A^\pi(s,a)$ is used instead of the reward $r(s,a)$ in order to lower the variance of the gradient g_{aug}. The changes in the magnitude of the advantage function $A^\pi(s,a) = Q(s,a) - V(s)$ will influence the ratio between the two summands and the normalisation is necessary of either of the change.
In order to keep the magnitude of the advantage function and control the gradient variance during training, the reward normalisation has been proposed. We follow the work by van Hasselt et al. [vGG+16].
We apply proposed value target normalisation from their work to normalise the rewards according to the formula:

$$r_{\text{norm}}(s,a) = \frac{r(s,a) - \mu_t}{\sigma_t}, \tag{4.2}$$

where

$$\mu_t = \frac{1}{N}\sum_i^N r(s,a)^{(i)} \quad \text{and} \quad \sigma_t^2 = \frac{1}{N}\sum_i^N \left(r(s,a)^{(i)} - \mu_t\right)^2, \tag{4.3}$$

at time step t.
These formulas can be generalised to online case by the introduction of the step size $\beta \in [0,1]$:

$$\mu_t = (1-\beta)\mu_{t-1} + \beta r(s,a) \quad \text{and} \quad \nu_t = (1-\beta)\nu_{t-1} + \beta r(s,a)^2, \tag{4.4}$$

with ν_t being the second moment. The estimated standard deviation is:

$$\sigma_t^2 = \nu_t - \mu_t^2. \tag{4.5}$$

This yields the exponential moving average which puts more weight on the recent data points as β step size is constant. The initial values of μ_0 and σ_0 are set arbitrarily to 0 and 1 respectively.
The formulation which suits better to our setup with the samples acquired in batches is:

$$\mu_t = (1-\beta)\mu_{t-1} + \beta\mu_{\text{cur}} \quad \text{and} \quad \sigma_t = (1-\beta)\sigma_{t-1} + \beta\sigma_{\text{cur}}, \tag{4.6}$$

where μ_{cur} and σ_{cur} are mean and standard deviation for the current batch calculated according to (4.3).
The results of reward normalisation are presented in Section 5.3.

4.2 Sample augmentation

A commonly used technique in deep learning to improve the network accuracy is the data augmentation [SK19]. In case of image classification, the images are subjected to random transformations, such as resizing, rotation, brightening, in order to achieve better generalisation to new examples using the existing dataset.

The similar approach has been considered for the deep reinforcement learning. Here the augmentation requires the knowledge of the task and careful implementation, since is has to consider the peculiarities of the given input observations from the environment. In our case only the rotation of the scene was a possible choice.
Sample augmentation is done according to the rule:

$$s_{t,\text{aug}} = \boldsymbol{R}(\theta) s_t, \tag{4.7}$$

where $\boldsymbol{R}$ is a rotation matrix around z axis:

$$\boldsymbol{R}(\theta) = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

and s_t is a $\mathbb{R}^{3xN}$ matrix consisting of stacked N indices of s_t which should be augmented.
Augmented elements of observations (see 2.2.1):

- position and orientation of hand base
- $\overrightarrow{p_O p_H}$ (vector between the object and hands' palm)
- $\overrightarrow{p_T p_H}$ (vector between the target and hands' palm)
- $\overrightarrow{p_T p_O}$ (vector between the target and object)

Other elements of observations are copied to the augmented observation without any change and then the augmentation of actions follows. The example of augmentation is provided in Figure 4.1 and the results of the evaluation are provided in Section 5.4.

4.3 Exploration enhancement

Exploration is widely researched subject in reinforcement learning [Tok10, IYY02, THF+17]. It often boils down to an exploitation-exploration problem, which is an open problem for machine learning approaches. In general, successful exploration strategy will allow the discovery of the high reward regions preventing falling in the suboptimal local minima.
The max-ent reinforcement learning, described in Section 3.2.2, lends a natural way of steering the exploration by adjusting the entropy weight ω in the learning objective (3.12).
Another approach to improve entropy is the introduction of the temperature term, which increases the policy variance by given rate. The variance augmentation is calculated according to the formula:

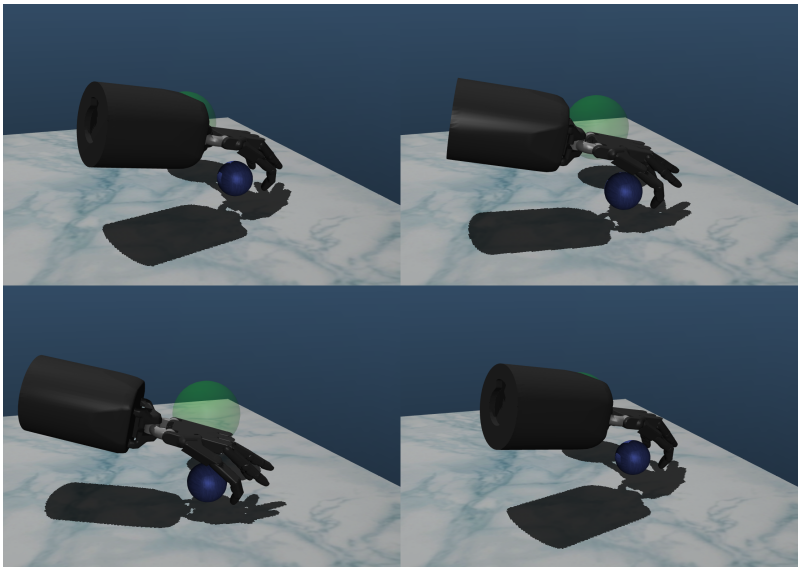

Figure 4.1: An example of a single augmented environment state. The rotation around z axis is presented from identical camera position.

$$\log \sigma = \log \sigma \cdot (1 + \text{current temp}) \tag{4.8}$$

where current temp decreases with each iteration i:

$$\text{current temp} = 0.95^{(i)} \cdot \text{base temp}, \quad i = \text{iteration no.} \tag{4.9}$$

Because of insufficient improvement comparing to the standard IRL method, the evaluation of the method was not included in the report. The structured exploration e.g. [GCM+19] could be considered instead increasing the randomness of the actions to learn more robust reward functions in consistent way.

4.4 Masking of the state space

Utilised inverse reinforcement learning approach, which in form resembles generative adversarial networks, seeks to solve the problem by mining the negative examples of actor states. However, due to the large input area, it is still susceptible to exploitation in case of examples out of explored state distribution.
The problem of lack of coverage of the input space to the reward function approximator can be seen as an example of adversarial attack on the artificial neural network representing the policy. Adversarial attacks on deep learning systems, such as this in Figure 4.2, is a well studied phenomenon, which hinders the broad application of deep learning systems. Here we follow the view of adversarial examples as in work by Goodfellow et al. [GPH+17] as "inputs to machine learning models that an attacker has intentionally designed to cause the model to make a mistake". In

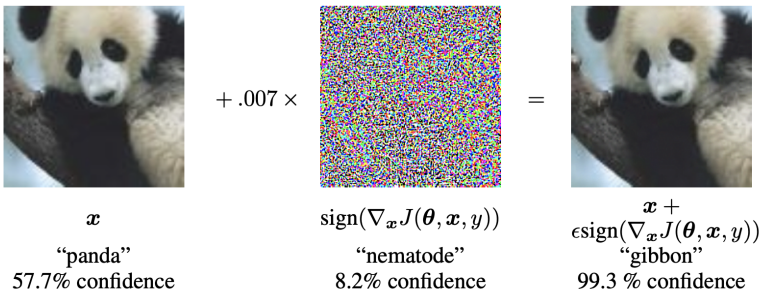

Figure 4.2: A demonstration of an adversarial example generation applied to GoogLeNet on ImageNet. [GSS15].

our case the mistake of reward function provides high reward $r(s, a)$ for state and action pair, which does not comply with the expected behaviour.
Similar aspect has been also observed in deep reinforcement learning in various framework. In Q-learning [GXL+18] the correct $Q(s, a)$ values cannot be learned from limited number of samples from the demonstrations, especially in high-dimensional input space or in IRL [FSG+18], where the shortcomings of the binary classifier are easily exploited by the actor. They alleviate this limitation by training only on the goal positions instead of the whole trajectory and mining the negative examples in the generative manner. This is not a viable solution as we are working with the demonstrations consisting of whole trajectories.

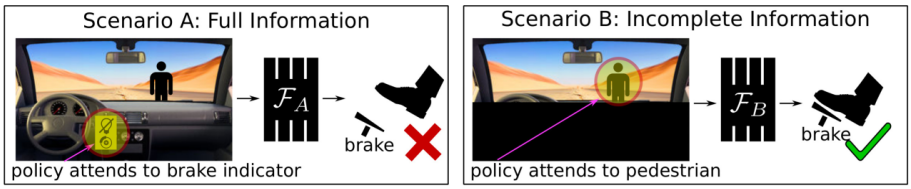

Figure 4.3: Example of causal confusion, where too much information yields worse imitation learning performance. [dJL19].

Another work suggests that too rich state information may cause a causal confusion where the cause of the action may be difficult to distinguish from its effect in the imitation learning setup yielding worse performance [dJL19] as presented in Figure 4.3. de Haan et al. propose to allow the expert queries to allow the information disambiguation.
The vulnerability of the learned policies in multi-agent environments has been described in work by Gleave et al. [GWKR19]. They have suggested to mitigate this effect by masking the observation to lower the dimensionality of the policy network. We propose to apply similar principle to IRL - lowering the input space to salient dimensions significantly improves the learning speed of the IRL algorithm and improves the robustness of the learned reward function as presented in Section 5.7.

This is achieved according to the same rule across different tasks - we mask the features of the hand in the input space to the reward function and keep dimensions specific to the MDP. The manual work for each new tasks is kept at minimum.

Chapter 5

Evaluation

The methods introduced in Chapter 4 have been evaluated in the task of object relocation with MuJoCo physics engine [TET12]. The goal of the task is the grasping of the object and its relocation to the given target position with a robotic Shadow Robot Dexterous Hand model (see Section 2.2). The simulation environment used in the experiments is depicted in Figure 5.1.

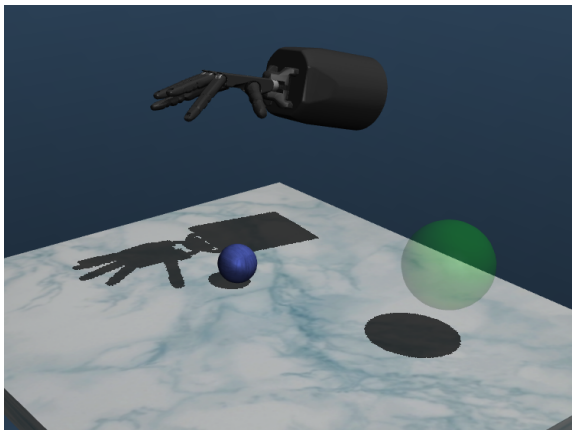

Figure 5.1: The visualisation of the task setup. The goal is to move the blue ball to the target location marked green.

The results are based on the success rate of the policy from fixed set of initial state s_0 of the environment. The episode is identified as successful if the object stays in the target location for at least 25 time steps, what corresponds to at least 12.5% of the episode duration.
The following elements have been evaluated:

- Particle swarm optimisation - the contribution towards higher efficiency for the demonstration acquisition and evaluation of the control types - Section 5.1
- State-of-the-art IRL algorithms - comparison performed to recognise the po-

tential utility of current SoA methods for dexterous hand manipulation - Section 5.2

- Reward normalisation - normalisation, which contributes to more stable augmented gradient - Section 5.3
- Sample augmentation - evaluation of how the extension of data contributes to training efficiency - Section 5.4
- Masking of the state space - aid to lack of robustness of the learned reward function - Section 5.5
- Transferability of the method - presents how the proposed method contributes to wider application of reinforcement learning for robotic dexterous hand manipulation - Section 5.6

5.1 Particle swarm optimisation evaluation

Particle swarm optimisation was used in the previous work for grasping dexterous hand manipulation before [AGK18]. We follow the similar principle and perform the evaluation in the object relocation task to evaluate the influence of the demonstration optimisation on the efficiency of learning.

Influence of the demonstration refinement on the task success rate

The successful execution of the task using the hand pose retargeting and motion retarget in the simulator is very challenging, especially using the position control. Empirically we were able to capture approximately 2.5 more successful demonstrations if the particle swarm optimisation was used during the motion retargeting when using position control. The velocity control has performed much better in the simulation environment providing approximately 8 times more successful demonstrations.
To reflect the difficulty of the demonstrations execution and evaluate how it corresponds with the learning capabilities, we have captured 3 demonstrations using position control without PSO, 8 demonstrations with position control and PSO and 24 demonstrations with the velocity control. Those demonstrations were used for direct learning with DAPG algorithm.

Reinforcement learning with refined demonstrations

The results of the evaluation are depicted in Figure 5.2.
While the superiority of the velocity control in our simulation environment accounts for many more demonstrations for the agent, the position control with PSO vastly outperforms the velocity control in terms of the learning speed with reinforcement

learning. We hypothesise that it is due to necessity to provide the control input type, joint velocities, in the state space besides the joint positions vector, which vastly increases the policy input dimension from 39 (see 2.2.1) to 63. This indicates, that the position control with PSO is the optimal way of learning dexterous hand manipulations from demonstrations.
Nevertheless, even though the position control with particle swarm optimisation performed best in this task (using Shadow Robot Dexterous hand for which the demonstration acquisition system described in Section 2.3 has been created), the learning speed is a magnitude slower in our MDP than in the equivalent task (Adroit) provided by Rajeswaran et al. [RKG+17] (compare with Figure 5.3). In order to allow efficient learning and have apples to apples comparison with state-of-the-art dexterous hand manipulation method by Rajeswaran et al. [RKG+17] it was decided to work on the object relocation in their MDP, provided with 25 demonstrations.

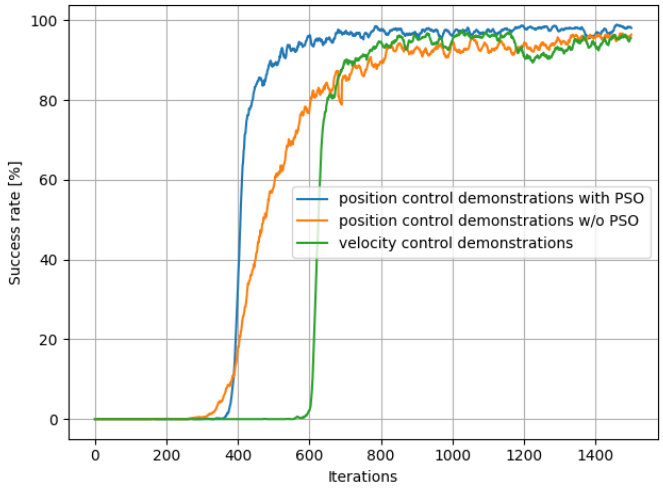

Figure 5.2: Evaluation of the particle swarm optimisation depending on the input type. The velocity control allows much higher success rate than position control, but the addition of the PSO increases the success rate providing the additional samples, which allow to outperform the velocity control.

5.2 State-of-the-art IRL algorithms

Prior to starting the work on improvement of inverse reinforcement learning the state-of-the-art inverse reinforcement learning algorithms had to be evaluated for the application in dexterous hand manipulation.
Figure 5.3 depicts the results of performance comparison of 3 state-of-the-art imitation learning methods, which were introduced previously in Section 3.2.1. The tested algorithms:

- GCL - Guided Cost Learning [FLA16]
- GAIL - Generative Adversarial Imitation Learning [HE16]
- AIRL - Adversarial Inverse Reinforcement Learning [FLL17]

In the experiments we have used the default hyperparameters delivered with the source code of the methods [Fu18]. The hyperparameters with their values are listed in Appendix A.

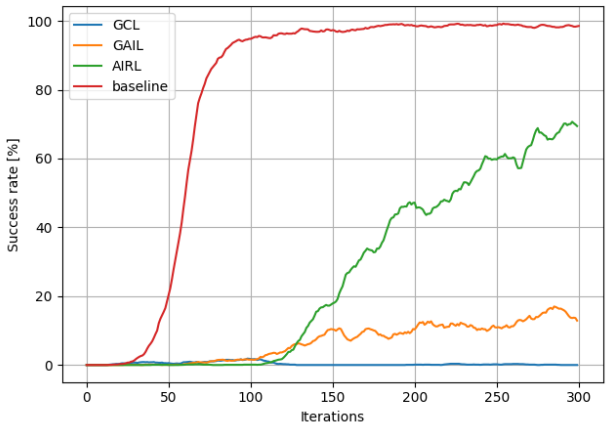

Figure 5.3: Comparison of 3 state-of-the-art inverse reinforcement learning algorithm together with the results of the baseline method for reference. The AIRL outperforms the 2 other IRL methods when learning the reward function, but is still significantly less sample efficient than the baseline method.

The results indicate the supremacy of the adversarial inverse reinforcement learning [FLL17]. This method also aims for the robustness to the changes of the environment, what should improve the learned reward making it especially appealing for the high dimensional task.

5.3 Reward normalisation

The non-stationarity of the reward function may confuse the policy learning [TGR18] and disturb the balance in the formulation of the used augmented gradient 4.1. In order to avoid it, the reward normalisation has been proposed. The results of its application are presented in Figure 5.4. They confirmed the importance for joint learning of reward and the policy.
The updates of μ_t and σ_t according to (4.6) in each discriminator step significantly perturb the optimisation process in case of too high normalisation learning rate β, but the proper value rises the quality of the learned policy (normalisation learning rate of 0.005) if compared to no reward normalisation (learning rate of 0).

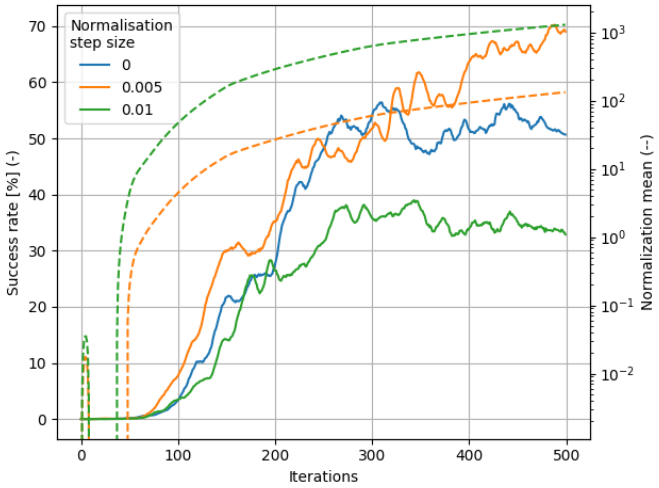

Figure 5.4: Evaluation of reward normalisation with respect to adjusted the normalisation steps size β used as in (4.6). The success rate is depicted with continuous line, the normalisation mean μ_t with dashed line in logarithmic scale. The normalisation allows the convergence to higher success rate.

5.4 Sample augmentation

The sample augmentation from Section 4.1 has been performed with the rotation angles sampled from the uniform distribution between -25° and 25° according to the method described in Section 4.2.
The results of the experiments of sample augmentation are depicted in Figure 5.5. The data is augmented 5 times, what is equivalent to collecting one new trajectory set. This significantly improves data acquisition rate and results in better training speed overall. It remains to be seen if such a method could be considered for real-world direct RL, because we can assume that the automatic augmentation could be unable to represent distinct non-linearities, which would characterise the environment.

5.5 Masking of the state space

Masking of the input space have been the solution to adversarial attacks in reinforcement learning, as described in Section 4.4. Here the method have been tested for the inverse reinforcement learning framework. Based on the prior knowledge of the MDP we select the last 9 dimensions of the input space as the input to the reward function approximator, which are sample specific and mask first 30 dimensions of state space. The use of only the subset of features for the reward function network lowers the possibility of observing off-distribution samples, which would lower the robustness of the inferred reward function as demonstrated empirically in Section

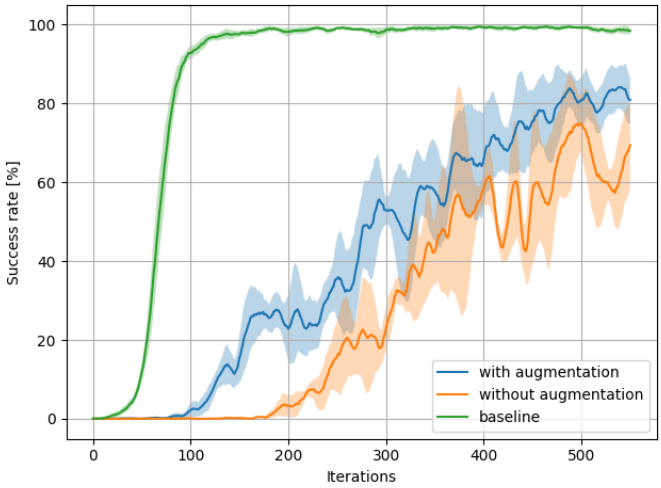

Figure 5.5: Evaluation of sample augmentation for IRL. The graph presents that the use of data augmentation for IRL improves the learning performance. The results of direct learning with the baseline method have been attached.

5.7.

The selected last 9 dimensions are the 3 vectors spanned between 3 points:

- p_H position of point below the middle of hand's palm
- p_O position of object
- p_T position of target location

according to specification of the state space in Section 2.2.1.
The results of IRL with masked state space are depicted in Figure 5.6.

5.6 Transferability of the method

The prevention of manual specification of reward function would be an advantage over the state-of-the-art dexterous hand manipulation if the extensive labour work on the manual reward specification in new tasks could be avoided [CLB+17].
In order to demonstrate the transferability of our method different hand manipulation tasks have been used for the evaluation: in-hand object manipulation (Figure 5.7) and tool usage (Figure 5.8). These MDPs have been used to compare proposed IRL method to the state-of-the-art dexterous hand manipulation method [RKG+17] with manually defined reward function and state-of-the-art inverse reinforcement learning. Because of higher dimension of the state space than in the original object relocation task, we provide 50 expert demonstrations instead of 25 demonstrations

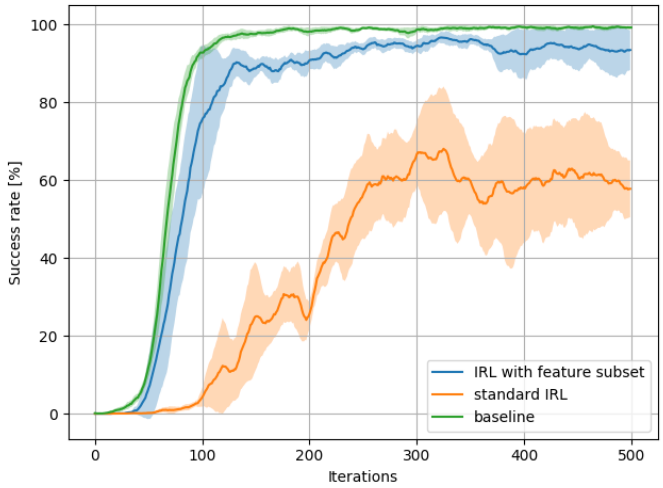

Figure 5.6: The results of the AIRL algorithm learning with the input state space masked to the last 9 dimensions comparing to the original input space and baseline. The performance with feature masking is similar the baseline, but does not converge to its high success rate. One standard deviation between evaluation runs is marked with shaded colour for each line.

to improve the learning capacities for evaluation of both methods: state-of-the-art IRL and our extension proposal.

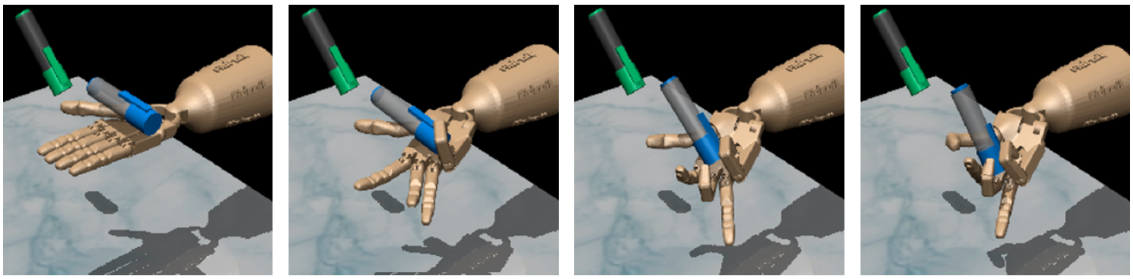

Figure 5.7: In-hand manipulation task: the goal of the task is the rotation of the object to the desired position marked by the green pen in the fixed dexterous hand. [RKG+17]

For the evaluation of transferability we used the masking of the state-space and addition of the random noise samples. We mask the first 30 dimensions, exactly as in the object relocation task, and leave the last dimensions, which are task specific. The results presented in Figure 5.9 prove that our method allows the high transferability without any additional labour while the baseline requires the engineering of the reward function for each task separately. This shows the higher versatility of our method and clear advantage over SoA dexterous hand manipulation approach [RKG+17].

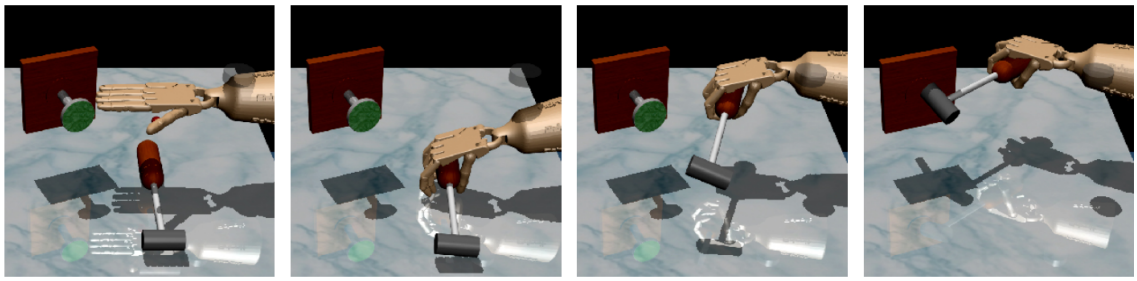

Figure 5.8: Tool usage task: The hammer has to be picked from the table and used to hit the nail into board with significant force. Task is successful if the whole nail is placed into board. The environment allows the force exertion only from the provided tool. [RKG+17]

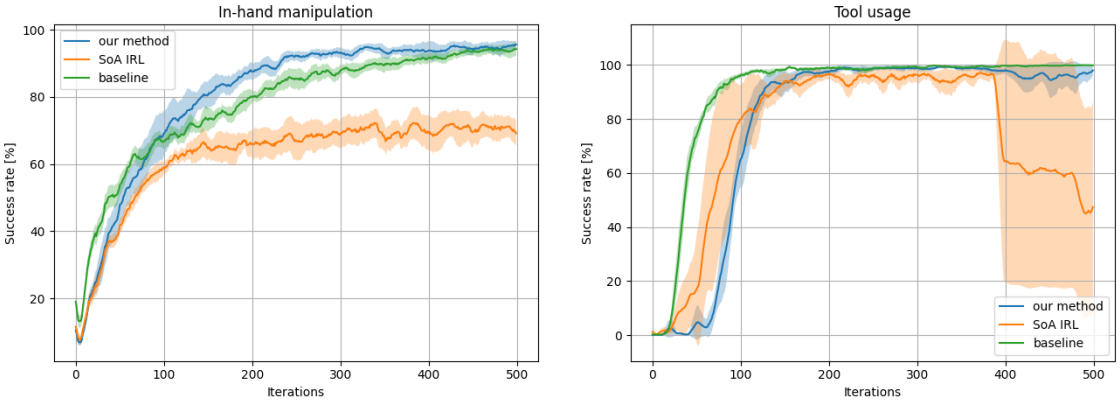

Figure 5.9: The proposed method is able to adapt to new setup using just the provided demonstration and delivers results comparable with the results of direct learning where the manual reward function specification is necessary.

5.7 Robustness of the reward function

To understand the reason for the superiority of our method in the range of dexterous hand manipulation tasks we quantitatively evaluate the robustness of the learned reward function.

We use the fixed learned reward function obtained at the end of the learning process to teach a new policy from scratch. The results of the evaluation in the tasks are presented in Table 5.1.

Our method consistently outperforms the state-of-the-art IRL (vanilla IRL) [FLL17], what proves that we obtain more robust reward function, which allow better learning. It naturally implies better efficiency when learning policy in the GAN setup. This claim was supported by the qualitative assessment of the reward function in Section 6.1.

	Object relocation	Tool usage
Vanilla IRL [FLL17]	-2.03	-229.22
Our method w/o noise samples	148.73	1404.31
Our method with noise samples	**186.35**	**6125.23**

Table 5.1: Results of learning from scratch on the fixed reward function taken at the end of IRL learning process. The values are the average maximum returns from the original reward function over 10 training iterations.

Chapter 6

Discussion

6.1 Robustness of the learned reward function

Learning of the reward jointly with the policy using the proposed methods allows the transfer of the method to novel task with minimal labour what is a considerable improvement over state-of-the-art methods for dexterous hand manipulation.
But during the work on the inverse reinforcement learning we devoted time to evaluate the learned reward functions for the ability to use the for learning from scratch (see evaluation results in Figure 6.1). Even though we are able to learn challenging tasks without manual feature engineering, the fixed reward functions are not suitable for learning from scratch. This is in line with the prior findings for GAN [GPM+] and shows, that joint learning is the applicable solution for dexterous hand manipulation.
The results of state-of-the-art work [CLB+17, FSG+18] suggest that the reward function is not suitable for robust learning and transfer, because of deficiencies of the learned reward function in high dimensional space.
In the interest of getting an important insight into understanding the reason for the lack of robustness of the learned reward function in dexterous hand manipulation, the visualisation of the learned reward has been created (Figure 6.1). Clearly, the reward function of SoA IRL is not suitable for the robust learning from scratch because of the high reward areas outside of the target. Our method learns the reward function, which guides to the region of high reward to the target, what makes it superior in the GAN-IRL setting.

6.2 Combination of proposed IRL methods

The respective proposed methods have improved the learning speed of IRL for dexterous hand manipulation. Especially significant improvement was observed with masking of the state space and reward function normalisation. A natural direction is the combination of all the methods to achieve superior performance to all

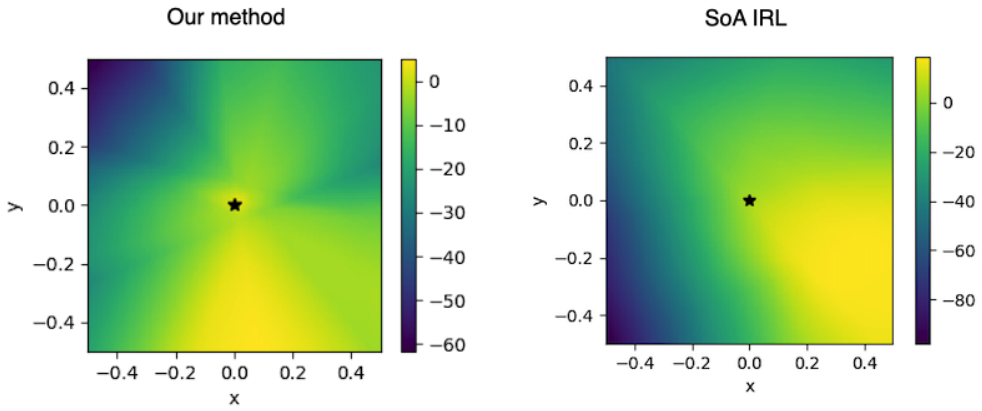

Figure 6.1: Plot of the learned reward function in object relocation task. It presents the learned reward value depending on the distance between object and target given in meters. Target position is marked with a star. Our method learns high reward for the target position as expected, while SoA IRL [FLL17] provides higher values to the areas outside goal position.

the respective ones. This was not observed, the results deteriorated if the methods were combined. The possible reason for it is that the distinct elements may not be complementary and an additional study would be required to understand how they can be fruitfully combined. Additionally, from the practical standpoint high number of hyperparameters to optimise may hinder the practical application. E.g. there have been multiple improvements proposed to Deep Q-Networks [MKS+13, MKS+15, VHGS16, WSH+16, SQAS15] and we could expect that if combined, they should improve performance as presented in the following work called Rainbow [HMVH+18], but for the practical application simpler methods such as Soft Actor-Critic [HZH+19] are often preferred.
For this reason the analysis of the combination of proposed methods for IRL is left for the future work.

Chapter 7

Conclusion

7.1 Conclusion

We have considered the problem of lack of transferability of state-of-the-art dexterous hand manipulation approach [RKG+17] caused by requirement to handcraft the reward function in each task to obtain expected behaviour. Reward engineering is both tedious and potentially dangerous especially for robotics, because of the likely reward function misspecification [CA16, AOS+16]. We proposed methods, which allow application of inverse reinforcement learning for efficient learning of challenging dexterous hand manipulation tasks directly from the human demonstrations. It is an important step toward higher applicability of reinforcement learning in robotics by minimising the required manual work and making the framework more versatile, while keeping the learning efficient.
We have evaluated the task with respect to the state-of-the-art reinforcement learning methods [FLL17, HE16] and the direct learning methods with manually specified reward function [RKG+17]. It vastly outperforms current inverse reinforcement learning methods by lowering the state space to the relevant dimensions and improves the robustness of the reward function by the addition of noise samples, normalisation and sample augmentation. Our method shows the superiority in terms of the transferability to new MDPs by learning the robust reward functions in demanding dexterous hand manipulation tasks.

7.2 Future work

Due to the time constraints of the master's thesis the method was not evaluated with the actual hardware and is left for the future work. There are several methods by which the policy transfer real world is possible [LKD+18, TFR+17, OPN17, JWK+] and the success of the zero-shot transfer using the domain randomisation technique for the dexterous hand manipulation [OAA+19] make us optimistic about the potential success of the method.

Secondly, the combination of the proposed methods did not deliver better evaluation results comparing to results obtained for each component used in isolation. Additional analysis could give us understanding of the relation between each extension and could achieve better performance of IRL for dexterous hand manipulation.
Thirdly, Rajeswaran et al. [RKG+17] claimed, that their policy-gradient based method has unparalleled performance in challenging high-dimensional tasks of dexterous hand manipulation. But considering the recent improvements in the efficient state-of-the-art off-policy reinforcement learning one could wonder if the novel methods such as Soft Actor-Critic [HZH+19] could potentially deal with large dimensions of action and state spaces while being more sample efficient. If one of efficient off-policy approaches would be coped with our IRL methods for dexterous hand manipulation it could potentially outperform the baseline method [RKG+17] both with respect to transferability and efficiency making the method even more compelling for robotics.

Appendix A

Implementation details

A.1 Physics engine and simulation environments

For the evaluation we used MuJoCo physics engine [TET12] with universal OpenAI Gym framework [BCP+16]. The simulation tasks have been adopted from the work by Rajeswaran et al. [RLTK17] from the corresponding GitHub repository.

A.2 Learning algorithm hyperparameters

We have used fixed hyperparameters for each of the tasks with exception for in-hand manipulation task, where 1 epoch has been used for behaviour cloning.

Behaviour cloning:

- epochs: 5
- learning rate: 1e-3

Policy (actor):

- number of FC layers: 2
- units per layer: 32
- activation function: Tanh
- step size: 0.1
- gamma: 0.995
- GAE lambda: 0.97
- trajectories per update: 200

Value function (critic):

- epochs: 2
- batch size: 64
- learning rate: 1e-3

Discriminator:

- number of FC layers: 2
- units per layer: 64
- activation function ReLU
- learning rate: 1e-3
- batch size: 256
- trajectories per update: 200
- max updates of generator per discriminator update: 4
- steps till max update no. for generator: 150
- minimal loss threshold: 0.01

Noise samples:

- percentage in all samples: 20%
- standard deviation: 1.0

A.3 Particle swarm optimisation hyperparameters

- particles number: 16
- optimisation steps: 5
- c_1 (see (3.14)): 2.8
- c_2 (see (3.14)): 1.3
- convergence threshold: 5e-05
- distance threshold for PSO activation: 0.075

List of Figures

Acronyms and Notations

AIRL Adversarial Inverse Reinforcement Learning

DAPG Demo Augmented Policy Gradients

DRL Deep Reinforcement Learning

GAIL Generative Adversarial Imitation Learning

GAN Generative Adversarial Networks

GCL Guided Cost Learning

HPE Hand Pose Estimation

IRL Inverse Reinforcement Learning

MDP Markov Decision Process

PPO Proximal Policy Optimisation

PSO Particle Swarm Optmisation

RL Reinforcement Learning

SoA State of the art

Bibliography

[AGK18] Dafni Antotsiou, Guillermo Garcia-Hernando, and Tae-Kyun Kim. Task-Oriented Hand Motion Retargeting for Dexterous Manipulation Imitation. *arXiv:1810.01845 [cs]*, October 2018.

[AN04] Pieter Abbeel and Andrew Y. Ng. Apprenticeship learning via inverse reinforcement learning. In *Twenty-First International Conference on Machine Learning - ICML '04*, page 1, Banff, Alberta, Canada, 2004. ACM Press. doi:10.1145/1015330.1015430.

[AOS+16] Dario Amodei, Chris Olah, Jacob Steinhardt, Paul Christiano, John Schulman, and Dan Mané. Concrete Problems in AI Safety. *arXiv:1606.06565 [cs]*, July 2016.

[AZH+19] Michael Ahn, Henry Zhu, Kristian Hartikainen, Hugo Ponte, Abhishek Gupta, Sergey Levine, and Vikash Kumar. ROBEL: Robotics Benchmarks for Learning with Low-Cost Robots. *arXiv:1909.11639 [cs, stat]*, September 2019.

[BCP+16] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. OpenAI Gym. *arXiv:1606.01540 [cs]*, June 2016.

[BGN19] Daniel S. Brown, Wonjoon Goo, and Scott Niekum. Better-than-Demonstrator Imitation Learning via Automatically-Ranked Demonstrations. In *arXiv:1907.03976 [Cs, Stat]*, Osaka, October 2019.

[BS99] C. Breazeal and B. Scassellati. How to build robots that make friends and influence people. In *Proceedings 1999 IEEE/RSJ International Conference on Intelligent Robots and Systems. Human and Environment Friendly Robots with High Intelligence and Emotional Quotients (Cat. No.99CH36289)*, volume 2, pages 858–863, Kyongju, South Korea, 1999. IEEE. doi:10.1109/IROS.1999.812787.

[CA16] Jack Clark and Dario Amodei. Faulty Reward Functions in the Wild. https://openai.com/blog/faulty-reward-functions/, December 2016.

[CLB+17] Paul F Christiano, Jan Leike, Tom Brown, Miljan Martic, Shane Legg, and Dario Amodei. Deep Reinforcement Learning from Human Preferences. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 4299–4307. Curran Associates, Inc., 2017.

[DBP+16] Alexander Dietrich, Kristin Bussmann, Florian Petit, Paul Kotyczka, Christian Ott, Boris Lohmann, and Alin Albu-Schäffer. Whole-body impedance control of wheeled mobile manipulators: Stability analysis and experiments on the humanoid robot Rollin' Justin. *Autonomous Robots*, 40(3):505–517, March 2016. doi:10.1007/s10514-015-9438-z.

[dJL19] Pim de Haan, Dinesh Jayaraman, and Sergey Levine. Causal Confusion in Imitation Learning. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d\textquotesingle Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 11693–11704. Curran Associates, Inc., 2019.

[EH19] Tom Everitt and Marcus Hutter. Reward Tampering Problems and Solutions in Reinforcement Learning: A Causal Influence Diagram Perspective. *arXiv:1908.04734 [cs]*, August 2019.

[FASL18] Pietro Falco, Abdallah Attawia, Matteo Saveriano, and Dongheui Lee. On Policy Learning Robust to Irreversible Events: An Application to Robotic In-Hand Manipulation. *IEEE Robotics and Automation Letters*, 3(3):1482–1489, July 2018. doi:10.1109/LRA.2018.2800110.

[FCAL16] Chelsea Finn, Paul Christiano, Pieter Abbeel, and Sergey Levine. A Connection between Generative Adversarial Networks, Inverse Reinforcement Learning, and Energy-Based Models. *arXiv:1611.03852 [cs]*, November 2016.

[Fin] Chelsea Finn. Lecture slides.

[FLA16] Chelsea Finn, Sergey Levine, and Pieter Abbeel. Guided Cost Learning: Deep Inverse Optimal Control via Policy Optimization. *arXiv:1603.00448 [cs]*, March 2016.

[FLL17] Justin Fu, Katie Luo, and Sergey Levine. Learning Robust Rewards with Adversarial Inverse Reinforcement Learning. *arXiv:1710.11248 [cs]*, October 2017.

[FSG+18] Justin Fu, Avi Singh, Dibya Ghosh, Larry Yang, and Sergey Levine. Variational Inverse Control with Events: A General Framework for

Data-Driven Reward Definition. *arXiv:1805.11686 [cs, stat]*, November 2018.

[Fu18] Justin Fu. Inverse RL. https://github.com/justinjfu/inverse_rl, June 2018.

[GCM+19] Yijie Guo, Jongwook Choi, Marcin Moczulski, Samy Bengio, Mohammad Norouzi, and Honglak Lee. Self-Imitation Learning via Trajectory-Conditioned Policy for Hard-Exploration Tasks. *arXiv:1907.10247 [cs, stat]*, November 2019.

[GELA17] Abhishek Gupta, Clemens Eppner, Sergey Levine, and Pieter Abbeel. Learning Dexterous Manipulation for a Soft Robotic Hand from Human Demonstration. *arXiv:1603.06348 [cs]*, March 2017.

[GPH+17] Ian J. Goodfellow, Nicola Papernot, Sandy Huang, Rocky Duan, Pieter Abbeel, and Jack Clark. Attacking machine learning with adversarial examples, Feb 2017. URL: https://openai.com/blog/adversarial-example-research/.

[GPM+] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative Adversarial Nets. page 9.

[GSS15] Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and Harnessing Adversarial Examples. *arXiv:1412.6572 [cs, stat]*, March 2015.

[GWKR19] Adam Gleave, Michael Dennis Cody Wild, Neel Kant, and Sergey Levine Stuart Russell. Adversarial Policies: Attacking Deep Reinforcement Learning. page 11, 2019.

[GXL+18] Yang Gao, Huazhe Xu, Ji Lin, Fisher Yu, Sergey Levine, and Trevor Darrell. Reinforcement Learning from Imperfect Demonstrations. *arXiv:1802.05313 [cs, stat]*, February 2018.

[HE16] Jonathan Ho and Stefano Ermon. Generative Adversarial Imitation Learning. *arXiv:1606.03476 [cs]*, June 2016.

[HGEJ17] Ahmed Hussein, Mohamed Medhat Gaber, Eyad Elyan, and Chrisina Jayne. Imitation Learning: A Survey of Learning Methods. *ACM Computing Surveys*, 50(2):1–35, April 2017. doi:10.1145/3054912.

[HMVH+18] 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 *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[HVP+17] Todd Hester, Matej Vecerik, Olivier Pietquin, Marc Lanctot, Tom Schaul, Bilal Piot, Dan Horgan, John Quan, Andrew Sendonaris, Gabriel Dulac-Arnold, Ian Osband, John Agapiou, Joel Z. Leibo, and Audrunas Gruslys. Deep Q-learning from Demonstrations. *arXiv:1704.03732 [cs]*, November 2017.

[HZH+19] Tuomas Haarnoja, Aurick Zhou, Kristian Hartikainen, George Tucker, Sehoon Ha, Jie Tan, Vikash Kumar, Henry Zhu, Abhishek Gupta, Pieter Abbeel, and Sergey Levine. Soft Actor-Critic Algorithms and Applications. *arXiv:1812.05905 [cs, stat]*, January 2019.

[IYY02] Shin Ishii, Wako Yoshida, and Junichiro Yoshimoto. Control of exploitation–exploration meta-parameter in reinforcement learning. *Neural networks*, 15(4-6):665–687, 2002.

[JWK+] Stephen James, Paul Wohlhart, Mrinal Kalakrishnan, Dmitry Kalashnikov, Alex Irpan, Julian Ibarz, Sergey Levine, Raia Hadsell, and Konstantinos Bousmalis. Sim-To-Real via Sim-To-Sim: Data-Efficient Robotic Grasping via Randomized-To-Canonical Adaptation Networks. page 11.

[KGTL16] Vikash Kumar, Abhishek Gupta, Emanuel Todorov, and Sergey Levine. Learning Dexterous Manipulation Policies from Experience and Imitation. *arXiv:1611.05095 [cs]*, November 2016.

[KP09] Jens Kober and Jan Peters. Learning motor primitives for robotics. In *2009 IEEE International Conference on Robotics and Automation*, pages 2112–2118, Kobe, May 2009. IEEE. doi:10.1109/ROBOT.2009.5152577.

[KT15] Vikash Kumar and Emanuel Todorov. MuJoCo HAPTIX: A virtual reality system for hand manipulation. In *2015 IEEE-RAS 15th International Conference on Humanoid Robots (Humanoids)*, pages 657–663, Seoul, South Korea, November 2015. IEEE. doi:10.1109/HUMANOIDS.2015.7363441.

[LKD+18] Kendall Lowrey, Svetoslav Kolev, Jeremy Dao, Aravind Rajeswaran, and Emanuel Todorov. Reinforcement learning for non-prehensile manipulation: Transfer from simulation to physical system. *arXiv:1803.10371 [cs]*, March 2018.

[LL18] Shile Li and Dongheui Lee. Point-to-Pose Voting based Hand Pose Estimation using Residual Permutation Equivariant Layer. *arXiv:1812.02050 [cs]*, December 2018.

[MKS+13] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing Atari with Deep Reinforcement Learning. *arXiv:1312.5602 [cs]*, December 2013.

[MKS+15] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, February 2015. doi:10.1038/nature14236.

[MLS17] Richard M. Murray, Zexiang Li, and S. Shankar Sastry. *A Mathematical Introduction to Robotic Manipulation*. CRC Press, 1 edition, December 2017. doi:10.1201/9781315136370.

[NKLK19] Anusha Nagabandi, Kurt Konoglie, Sergey Levine, and Vikash Kumar. Deep Dynamics Models for Learning Dexterous Manipulation. *arXiv:1909.11652 [cs]*, September 2019.

[NMA+18] Ashvin Nair, Bob McGrew, Marcin Andrychowicz, Wojciech Zaremba, and Pieter Abbeel. Overcoming Exploration in Reinforcement Learning with Demonstrations. *arXiv:1709.10089 [cs]*, February 2018.

[NPAF15] Amy C Nau, Christine Pintar, Aimee Arnoldussen, and Christopher Fisher. Acquisition of visual perception in blind adults using the brainport artificial vision device. *American Journal of Occupational Therapy*, 69(1):6901290010p1–6901290010p8, 2015.

[OAA+19] OpenAI, Ilge Akkaya, Marcin Andrychowicz, Maciek Chociej, Mateusz Litwin, Bob McGrew, Arthur Petron, Alex Paino, Matthias Plappert, Glenn Powell, Raphael Ribas, Jonas Schneider, Nikolas Tezak, Jerry Tworek, Peter Welinder, Lilian Weng, Qiming Yuan, Wojciech Zaremba, and Lei Zhang. Solving rubik's cube with a robot hand. *arXiv preprint*, 2019.

[OAB+18] OpenAI, Marcin Andrychowicz, Bowen Baker, Maciek Chociej, Rafal Jozefowicz, Bob McGrew, Jakub Pachocki, Arthur Petron, Matthias Plappert, Glenn Powell, Alex Ray, Jonas Schneider, Szymon Sidor, Josh Tobin, Peter Welinder, Lilian Weng, and Wojciech Zaremba. Learning dexterous in-hand manipulation. *CoRR*, 2018. URL: http://arxiv.org/abs/1808.00177.

[OPN17] Takayuki Osa, Jan Peters, and Gerhard Neumann. Experiments with Hierarchical Reinforcement Learning of Multiple Grasping Policies.

In Dana Kulić, Yoshihiko Nakamura, Oussama Khatib, and Gentiane Venture, editors, *2016 International Symposium on Experimental Robotics*, volume 1, pages 160–172. Springer International Publishing, Cham, 2017. doi:10.1007/978-3-319-50115-4_15.

[PHL+] Ivaylo Popov, Nicolas Heess, Timothy Lillicrap, Roland Hafner, Gabriel Barth-Maron, Matej Vecerik, Thomas Lampe, Yuval Tassa, Tom Erez, and Martin Riedmiller. Data-efficient Deep Reinforcement Learning for Dexterous Manipulation. page 12.

[Pom89] Dean A Pomerleau. Alvinn: An autonomous land vehicle in a neural network. In *Advances in neural information processing systems*, pages 305–313, 1989.

[RKG+17] Aravind Rajeswaran, Vikash Kumar, Abhishek Gupta, Giulia Vezzani, John Schulman, Emanuel Todorov, and Sergey Levine. Learning Complex Dexterous Manipulation with Deep Reinforcement Learning and Demonstrations. *arXiv:1709.10087 [cs]*, September 2017.

[RLTK17] Aravind Rajeswaran, Kendall Lowrey, Emanuel V. Todorov, and Sham M Kakade. Towards Generalization and Simplicity in Continuous Control. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 6550–6561. Curran Associates, Inc., 2017.

[RPKS92] Aw Roe, Sl Pallas, Yh Kwon, and M Sur. Visual projections routed to the auditory pathway in ferrets: Receptive fields of visual neurons in primary auditory cortex. *The Journal of Neuroscience*, 12(9):3651–3664, September 1992. doi:10.1523/JNEUROSCI.12-09-03651.1992.

[RSF13] Eric Rohmer, Surya PN Singh, and Marc Freese. V-rep: A versatile and scalable robot simulation framework. In *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1321–1326. IEEE, 2013.

[SB18] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. Adaptive Computation and Machine Learning Series. The MIT Press, Cambridge, Massachusetts, second edition edition, 2018.

[Sha] Shadow Robot Company. *Shadow Dexterous Hand Technical Specification Shadow Dexterous Hand E Series*.

[SHKM92] Claude Sammut, Scott Hurst, Dana Kedzier, and Donald Michie. Learning to fly. In *Machine Learning Proceedings 1992*, pages 385–393. Elsevier, 1992.

[SHM+16] David Silver, Aja Huang, Chris J. Maddison, Arthur Guez, Laurent Sifre, George van den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, Sander Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, Timothy Lillicrap, Madeleine Leach, Koray Kavukcuoglu, Thore Graepel, and Demis Hassabis. Mastering the game of Go with deep neural networks and tree search. *Nature*, 529(7587):484–489, January 2016. doi:10.1038/nature16961.

[SK19] Connor Shorten and Taghi M Khoshgoftaar. A survey on image data augmentation for deep learning. *Journal of Big Data*, 6(1):60, 2019.

[SQAS15] Tom Schaul, John Quan, Ioannis Antonoglou, and David Silver. Prioritized experience replay. *arXiv preprint arXiv:1511.05952*, 2015.

[SSS+17] David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, Yutian Chen, Timothy Lillicrap, Fan Hui, Laurent Sifre, George van den Driessche, Thore Graepel, and Demis Hassabis. Mastering the game of Go without human knowledge. *Nature*, 550(7676):354–359, October 2017. doi:10.1038/nature24270.

[TET12] Emanuel Todorov, Tom Erez, and Yuval Tassa. MuJoCo: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5026–5033, Vilamoura-Algarve, Portugal, October 2012. IEEE. doi:10.1109/IROS.2012.6386109.

[TFR+17] 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. *arXiv:1703.06907 [cs]*, March 2017.

[TGR18] Aaron Tucker, Adam Gleave, and Stuart Russell. Inverse reinforcement learning for video games. *arXiv:1810.10593 [cs, stat]*, October 2018.

[THF+17] Haoran Tang, Rein Houthooft, Davis Foote, Adam Stooke, OpenAI Xi Chen, Yan Duan, John Schulman, Filip DeTurck, and Pieter Abbeel. # exploration: A study of count-based exploration for deep reinforcement learning. In *Advances in neural information processing systems*, pages 2753–2762, 2017.

[Tok10] Michel Tokic. Adaptive ε-greedy exploration in reinforcement learning based on value differences. In *Annual Conference on Artificial Intelligence*, pages 203–210. Springer, 2010.

[VBC+19a] Oriol Vinyals, Igor Babuschkin, Junyoung Chung, Michael Mathieu, Max Jaderberg, Wojtek Czarnecki, Andrew Dudzik, Aja Huang, Petko Georgiev, Richard Powell, Timo Ewalds, Dan Horgan, Manuel Kroiss, Ivo Danihelka, John Agapiou, Junhyuk Oh, Valentin Dalibard, David Choi, Laurent Sifre, Yury Sulsky, Sasha Vezhnevets, James Molloy, Trevor Cai, David Budden, Tom Paine, Caglar Gulcehre, Ziyu Wang, Tobias Pfaff, Toby Pohlen, Dani Yogatama, Julia Cohen, Katrina McKinney, Oliver Smith, Tom Schaul, Timothy Lillicrap, Chris Apps, Koray Kavukcuoglu, Demis Hassabis, and David Silver. AlphaStar: Mastering the Real-Time Strategy Game StarCraft II. https://deepmind.com/blog/alphastar-mastering-real-time-strategy-game-starcraft-ii/, 2019.

[VBC+19b] Oriol Vinyals, Igor Babuschkin, Wojciech M. Czarnecki, Michaël Mathieu, Andrew Dudzik, Junyoung Chung, David H. Choi, Richard Powell, Timo Ewalds, Petko Georgiev, Junhyuk Oh, Dan Horgan, Manuel Kroiss, Ivo Danihelka, Aja Huang, Laurent Sifre, Trevor Cai, John P. Agapiou, Max Jaderberg, Alexander S. Vezhnevets, Rémi Leblond, Tobias Pohlen, Valentin Dalibard, David Budden, Yury Sulsky, James Molloy, Tom L. Paine, Caglar Gulcehre, Ziyu Wang, Tobias Pfaff, Yuhuai Wu, Roman Ring, Dani Yogatama, Dario Wünsch, Katrina McKinney, Oliver Smith, Tom Schaul, Timothy Lillicrap, Koray Kavukcuoglu, Demis Hassabis, Chris Apps, and David Silver. Grandmaster level in StarCraft II using multi-agent reinforcement learning. *Nature*, 575(7782):350–354, November 2019. doi:10.1038/s41586-019-1724-z.

[vGG+16] Hado P van Hasselt, Arthur Guez, Arthur Guez, Matteo Hessel, Volodymyr Mnih, and David Silver. Learning values across many orders of magnitude. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems 29*, pages 4287–4295. Curran Associates, Inc., 2016.

[VHGS16] Hado Van Hasselt, Arthur Guez, and David Silver. Deep reinforcement learning with double q-learning. In *Thirtieth AAAI conference on artificial intelligence*, 2016.

[WSH+16] Ziyu Wang, Tom Schaul, Matteo Hessel, Hado Hasselt, Marc Lanctot, and Nando Freitas. Dueling network architectures for deep reinforcement learning. In *International conference on machine learning*, pages 1995–2003, 2016.

[ZBD08] Brian D Ziebart, J Andrew Bagnell, and Anind K Dey. Maximum Entropy Inverse Reinforcement Learning. page 1, 2008.

[Zie10] Brian D. Ziebart. *Modeling Purposeful Adaptive Behavior with the Principle of Maximum Causal Entropy.* PhD thesis, Carnegie Mellon University, December 2010.

License