Motor Synergy Emergence and Redundancy Quantification through Deep Reinforcement Learning in Robotics

Doctoral Thesis

Thesis Title

Motor Synergy Emergence and Redundancy Quantification

through Deep Reinforcement Learning in Robotics

Department of Robotics

Graduate School of Engineering

TOHOKU UNIVERSITY

JIAZHENG CHAI

Advising Professor at Tohoku Univ.	Professor Mitsuhiro Hayashibe
Research Advisor at Tohoku Univ.	–
Dissertation Committee Members Name marked with "○" is the Chief Examiner	○ Prof. Mitsuhiro Hayashibe 1 Prof. Satoshi Murata　2 Prof. Koichi Hashimoto (*Graduate School of Information Sciences*) 3 Dr. Juan Alvaro Gallego (*Imperial College London*)

TOHOKU UNIVERSITY

Graduate School of Engineering

Motor Synergy Emergence and Redundancy Quantification through Deep Reinforcement Learning in Robotics

(ロボティクスのための深層強化学習による
運動シナジー発現と 冗長性定量化に関する研究)

A dissertation submitted for the degree of

Doctor of Philosophy (Engineering)

Department of Robotics

by

Jiazheng CHAI

July 5, 2021

Motor Synergy Emergence and Redundancy Quantification through Deep Reinforcement Learning in Robotics

Jiazheng CHAI

Abstract

Synergy is a concept in the neuroscience field, wherein the central nervous system (CNS) uses a much smaller set of variables to control a large group of muscles to generate movements in biological systems, including humans. The co-activation of a set of muscles from a smaller number of neural commands, that is, motor synergies, can reduce the burden of the CNS significantly while imposing a certain level of coordination between joints that are closely related to each other for a certain movement. Essentially, motor synergies reduce the dimensionality in terms of the degrees of freedom of motion. These motor synergies are acquired by biological systems through natural growth and learning processes. Researchers in this field have been observing and analyzing existing motor synergies in biological systems, posing various hypotheses about the origin of the motor synergies in living organisms. Inspired by these studies, and in an attempt to further understand the motor synergy concepts behind the biological systems, the primary motivation in this thesis is to transit from the observation to creation of motor synergies.

To recreate the synergy emergence process, it is necessary to simulate the learning process of a biological system from its blank state to its functional state, that is, the state in which it is able to carry out a task efficiently using the learned motor synergies. The proposition in this thesis is to use simulated robotic agents to replace biological systems, and employ a learning algorithm, ideally an algorithm that shares certain similarities with the human learning process, to allow the robotic agent to learn to perform a task from scratch. Because the concerned robotic agents are highly redundant to better mimic biological systems, it is necessary to have a powerful learning algorithm to control the agents despite the redundancy.

The deep reinforcement learning (DRL) algorithm is the perfect choice for this thesis. First, the optimization process of a DRL algorithm is able to find the optimum control strategies in the infinite possible solution space of redundant robots.

Indeed, DRL has been proven to be a general algorithm that can be applied to a range of different robotic models and robotic tasks, and has been gaining attention in the field of robotics. More importantly, the reward system in DRL shares certain similarities with the biological goal-directed learning process. The synergy level during the learning phase of the DRL algorithm was then quantified to demonstrate the synergy emergence phenomenon. By attempting to recreate the synergy emergence process, it was desired to reveal the relationship between the motor synergies and the energy efficiency in the execution of a task. The findings from this study further provided the foundation for the subsequent two studies in this thesis, that is, the joint redundancy quantification study and the gait mode specification study, both having a strong relationship with the motor synergy concepts demonstrated in this thesis.

This thesis made three contributions, all revolving around the motor synergy concept and the DRL.

In the first study, the objective was to recreate the synergy emergence process in simulated agents using DRL algorithms, with the aim of analyzing the relationship between the motor synergy development and the energy efficiency of the agents throughout the learning process. Two state-of-the-art DRL algorithms were used to train several simulated robotic models with quadrupedal/half-quadrupedal structures, and their objective was to run forward as quickly as possible. Initially, the joints of the simulated agents were not sufficiently coordinated to perform a smooth running motion. As the learning progressed, the joints of the agents became more coordinated, or synergetic, and the running motion also improved accordingly. This phenomenon is referred to as the synergy emergence phenomenon. New synergy metrics were proposed that allow the measurement of the synergy level or the degree of coordination between joints during the DRL learning processes. The experimental results demonstrated that the running performance and energy efficiency of the agents increased as the synergy level steadily increased during learning. This synergy emergence phenomenon could be observed statistically in the learning agents, even though a synergy constraint has never been encoded into the reward function of the DRL algorithms. The proposed synergy-related metrics successfully distinguished the learning capability of the two DRL algorithms, suggesting that these metrics could be used as additional indices to evaluate DRL algorithms for motor learning. In addition, synergy analysis was also performed on a human-like robotic arm to show that the same results could be obtained for another type of robotic agent. It is also hoped that a DRL algorithm tends to determine solutions that are more synergetic than that by a PD controller.

As the second contribution of this thesis, a synergy metric proposed in the first part of this thesis, the Synergy Exploration Area (SEA) metric, was used as an alternative approach to quantify joint redundancy in redundant robotic agents. Various experiments were conducted with different robotic structures for different tasks, ranging from a simple robotic arm manipulation to a more complex robotic locomotion. The experimental results demonstrated that the SEA metric effectively quantified the relative joint redundancy over different robotic structures with varying degrees of freedom under unknown dynamic situations. The SEA metric could also quantify the kinematic and dynamic factors that affect the joint redundancy of a robotic agent. Indeed, the study successfully applied the concepts of the DRL optimization process to the quantification of redundancy in robotic agents, and the SEA metric acted as a bridge between these two domains. In addition, it was demonstrated that the SEA metric could be potentially used to evaluate the optimality of a robotic structure for a given task. This could help to select a better design for a robotic agent to ensure it had the appropriate degree of joint redundancy for an intended task, which would subsequently reduce the complexities of the final control method and prevent wasting resources by using an overly complex robot.

Finally, DRL was employed for quadruped gait generation for energetic analysis. To counter the fact that the DRL-trained agents do not necessarily possess a well-known gait type, a method was proposed to impose a certain gait type on the DRL-trained agents. The experimental results demonstrated that specifying a gait mode in the DRL learning process, it allowed a faster convergence of the learning process and synergy emergence process. This was because the mode specification reduced redundant solution space. More importantly, the proposed method successfully imposed a certain gait type on the quadrupeds, as opposed to the case without any gait specification. The advantages of using DRL were equally demonstrated as it could automatically exploit the physical condition of the robots, such as the passive spring effect between the joints during the learning process, similar to the adaptation skills of an animal. This study provides a framework for quadrupedal trot-gallop energetic analysis for different body structures, body mass distributions, and joint characteristics using DRL.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background

Synergy [1][2] is a concept in the neuroscience field, wherein the central nervous system (CNS) uses a much smaller set of variables to control a large group of muscles to generate movements in biological systems, including humans [3][4]. The co-activation of a set of muscles from a smaller number of neural commands, that is, motor synergies, can reduce the burden of CNS significantly [5] while imposing a certain level of coordination between joints that are closely related to each other for a certain movement. Essentially, motor synergies reduce the dimensionality in terms of the degrees of freedom of motion. These motor synergies are acquired by biological systems through natural growth and learning processes. Researchers in this field have been observing and analyzing existing motor synergies in biological systems [3][4][6][7], posing various hypotheses about the origin of the motor synergies in the living organisms. Inspired by these studies, and in an attempt to further understand the motor synergy concepts behind the biological systems, the primary motivation in this thesis is to transit from the observation to the creation of motor synergy, as shown in the Fig. 1.1.

To recreate the synergy emergence process, it is necessary to simulate the learning process of a biological system from its blank state to its functional state, that is, the state in which it is able to carry out a task energy efficiently using the learned motor synergies. The proposition in this thesis is to use simulated robotic agents to replace the biological systems, and employ a learning algorithm, ideally an algorithm which shares certain similarities with the human learning process, to allow the robotic agent to learn to perform a task from scratch. Because the concerned robotic agents are highly redundant to better mimic biological systems, it is necessary to have a

powerful learning algorithm to control the agents despite the redundancy.

The deep reinforcement learning (DRL) algorithm is the perfect choice for this thesis. First, the optimization process of a DRL algorithm is able to find the optimum control strategies in the infinite possible solution space of redundant robots. Indeed, DRL has been proven to be a general algorithm that can be applied to a range of different robotic models and robotic tasks [8][9], and has been gaining attention in the field of robotics [10][11]. More importantly, the reward system in DRL shares certain similarities with the biological goal-directed learning process [12][13][14][15]. The synergy level during the learning phase of the DRL algorithm was then quantified to demonstrate the synergy emergence phenomenon. By attempting to recreate the synergy emergence process, it was desired to reveal the relationship between the motor synergies and the energy efficiency in the execution of a task. The findings from this study further provided the foundation for the subsequent two studies in this thesis, that is, the joint redundancy quantification study and the gait mode specification study, both having a strong relationship with the motor synergy concepts demonstrated in this thesis.

In this section, the background of several concepts used in this thesis are introduced, including the motor synergy concept and its calculation, the joint redundancy, and the quadrupedal gait types. The knowledge of DRL necessary to understand this thesis is also explained in this section.

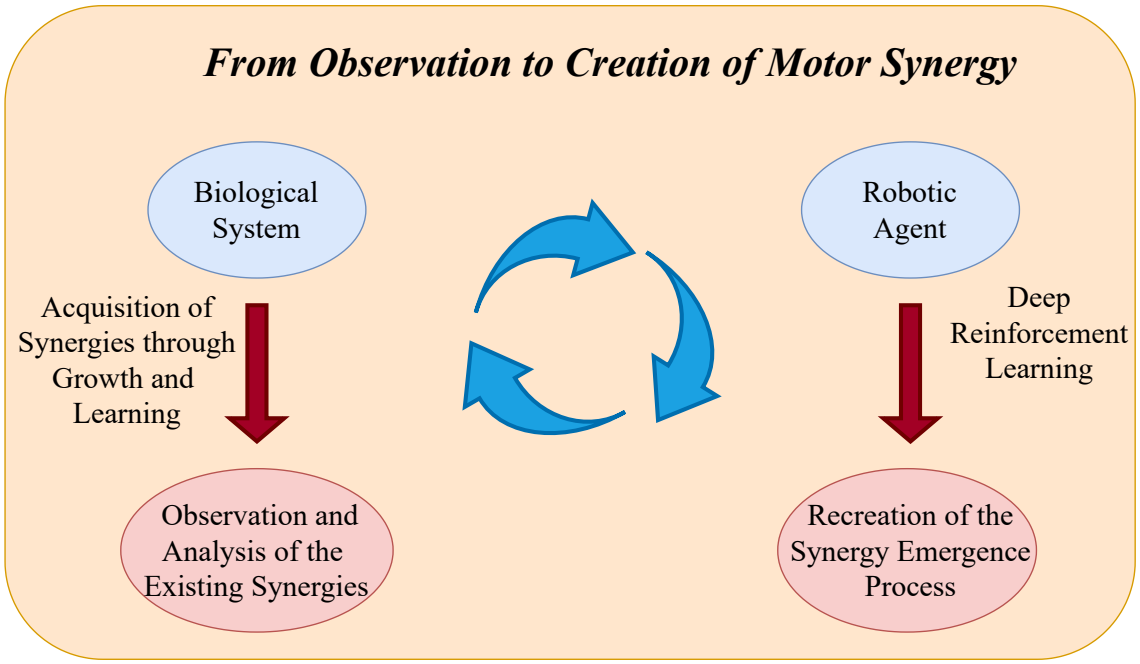


Figure 1.1: The primary motivation of the thesis. Taking inspiration from the study of motor synergy in biological systems, this thesis aims to recreate the synergy emergence process in robotic agents using deep reinforcement learning.

1.1.1 Motor synergy

As mentioned previously, motor synergy is a concept in the neuroscience field where the CNS has shown to use a much smaller set of variables to control a large group of muscles to generate movements in human. For illustrative purpose, Fig. 1.2 shows the motor synergy concept through a simplistic example of the CNS sending commands to a moving limb.

The authors of [16][17] formalized the calculation for extracting different types of motor synergy. The spatiotemporal synergy is first explained as it is mainly used in this thesis. One could also refer to [18] for a similar study of spatiotemporal synergy analysis. The spatiotemporal synergy has been proposed to model source

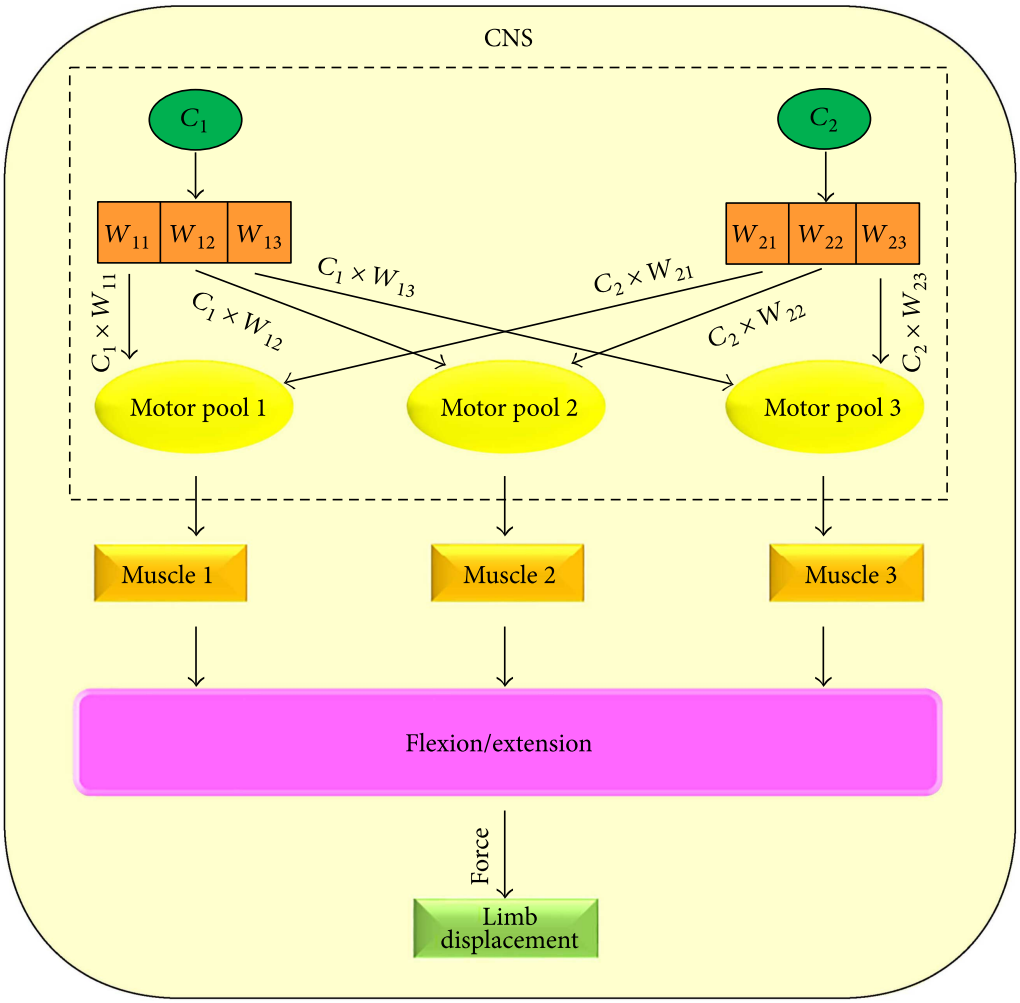


Figure 1.2: A graphical illustration of the motor synergy concept in a limb displacement task. The green components C_i represent the control signals from the CNS and the W_i represent the motor synergy components. The combinations of the control signals and the synergy components produce the final signals for the muscles responsible for the limb movements. The CNS typically sends fewer control signals C_i to control a large group of muscles for a certain task. [2]

signal pattern that is invariant in both space and time. The spatiotemporal synergy decomposition is as (1.1.1) where $x^l(t)$ indicates the source signals at time point t in trial number l and P is the total number of spatiotemporal synergy components. c_p^l are the mixing weights that change between trials while $w_p(t)$ is the p^{th} spatiotemporal synergy component, which is dependent on t but assumed to be invariant over trials.

$$x^l(t) = \sum_{p=1}^{P} c_p^l \cdot w_p(t) + residuals \tag{1.1.1}$$

The advantage of the spatiotemporal synergy decomposition is that it summarizes the space and time information in the source signals, hence it is more compact. Another type of synergy which is equally commonly used is the spatial synergy [16][19]. Spatial synergy models source signal pattern that is invariant in time and its decomposition is as (1.1.2), where the notations are the same as (1.1.1) except that the $c_p^l(t)$ is now dependent on t while w_p, which is the spatial synergy is constant over time. Fig. 1.3 gives a clearer idea of the spatial and spatiotemporal synergy extraction process of some sample signals. Fig. 1.4 illustrates in details the preprocessing methods of the raw collected signals for both the spatial synergy extraction and spatiotemporal synergy extracion.

$$x^l(t) = \sum_{p=1}^{P} w_p \cdot c_p^l(t) + residuals \tag{1.1.2}$$

If (1.1.1) and (1.1.2) are written in the matrix form and the *residuals* term are ignored, (1.1.3) is the simplified equation to be solved for both types of synergy, where X represents the source signals, W the synergy components matrix and C the corresponding activation matrix.

$$X = W \cdot C \tag{1.1.3}$$

One of the most commonly used algorithms to solve (1.1.3) is the principal component analysis (PCA). While the details of PCA are not presented here, the core idea is that there exists some underlying alternative axes that can replace the original axes of the data. The properties of these alternative axes are such that the first axis explains the most variance of the data, following by axes that explain less and less variance of the data. Figure 1.5 shows an example of the PCA

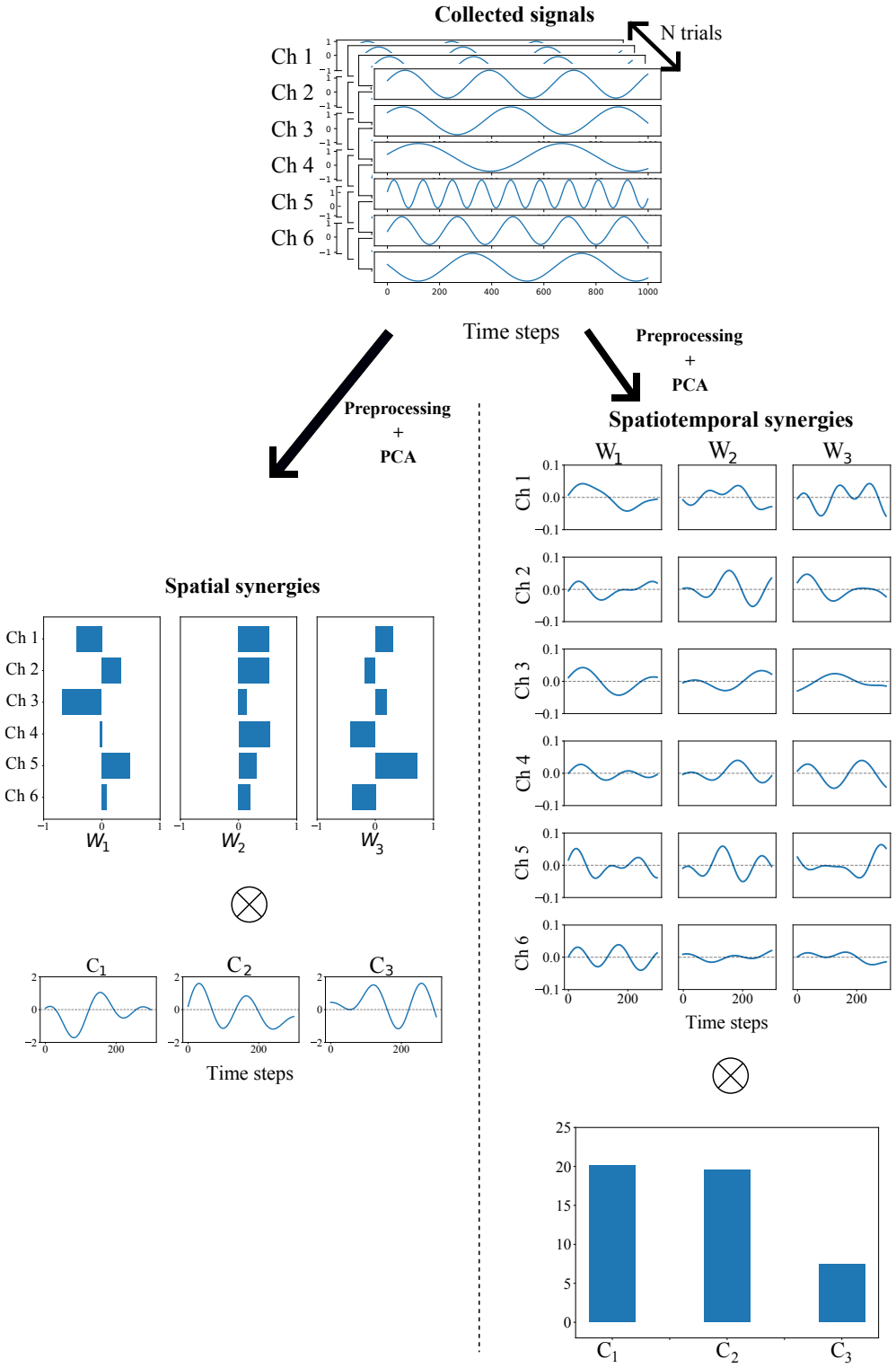


Figure 1.3: An illustration of the spatial (left) and spatiotemporal (right) synergies extraction process. On top of the diagram is the simulated signals with six channels and N trials are simulated. The signals are then preprocessed in preparation for the spatial or spatiotemporal synergies extraction. The processing method for each synergy extraction type is explained later in Figure 1.4. PCA is then used to decompose the preprocessed signals into synergy components W_i and the accompanying C_i components. The number of synergy components W_i can be varied during the PCA extraction process. In this example, three synergy components are extracted.

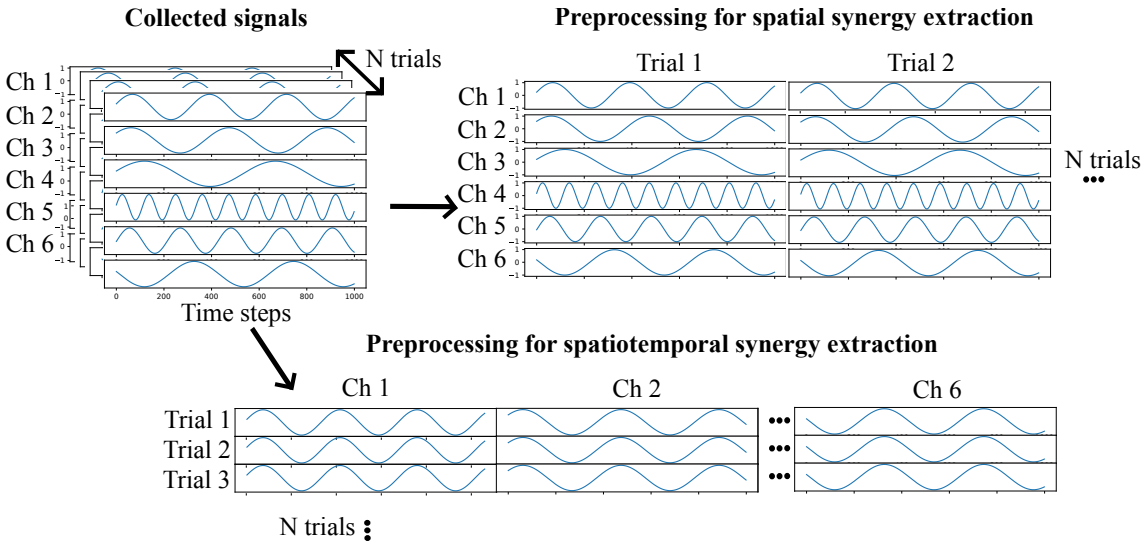


Figure 1.4: Preprocessing and reshaping of the input collected signals prior to the spatial synergy extraction (top right) and spatiotemporal synergy extraction (bottom). The collected signals consist of N trials, a certain length of time steps of truncated signals, and a certain number of channels, e.g. six channels in this example. For the spatial synergy extraction, the raw collected signals are preprocessed into a matrix where the vertical axis is the channels as shown on the top right of the figure. For the spatiotemporal synergy extraction, the raw collected signals are preprocessed into a matrix where the vertical axis is the trials as shown on the bottom of the figure.

components of the 2-dimensional data. As it can be easily noticed from the plot, the first PCA component explain the most variance of the data, followed by the second PCA component which explains less variance. While this example shows a simple 2-dimensional data, it can be easily scaled to very high-dimensional sparse data, and the first few PCA components of these data can typically explain most of the variance of the data. This is the main function of PCA, which is the dimension reduction for high-dimensional data that have high correlation between different dimensions. This is exactly what is needed in finding synergy components which can explain the highly correlated input signals. In simple explanations, the PCA algorithm finds the PCA components by minimizing the reconstruction error in (1.1.4) with respect to the source signals X by optimizing the C and W matrices, with $\|\cdot\|_F$ being the Frobenius Norm.

$$E^2 = \|X - W \cdot C\|_F^2 \tag{1.1.4}$$

The metric that is usually used to evaluate how well X can be represented from a certain W and its corresponding C is the coefficient of determination, or the R^2 metric. The R^2 can be calculated according to (1.1.5) and it ranges from zero to

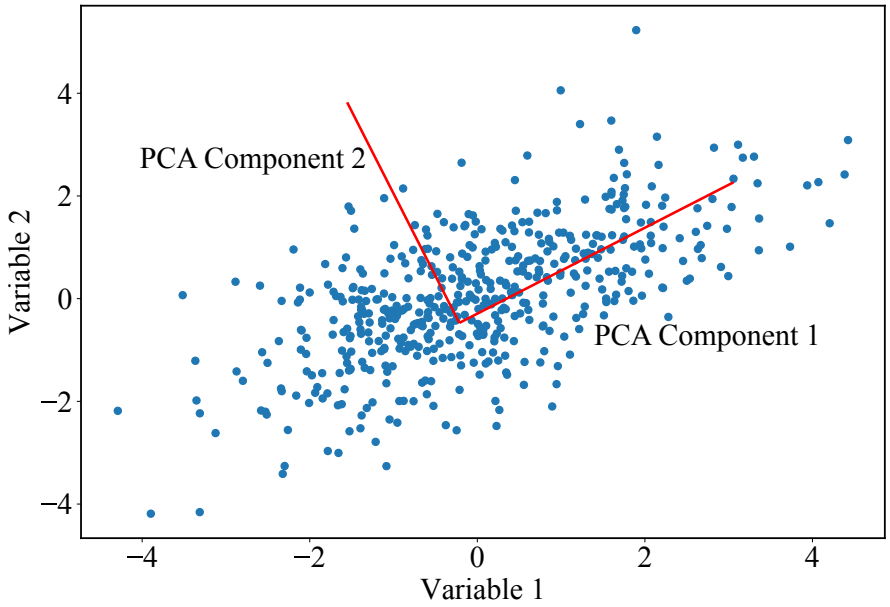


Figure 1.5: Plot showing some 2-dimensional data together with their first and second PCA components obtained using the PCA algorithm. The PCA components are found such that the first PCA component explains the most variance of the data, following by the second PCA component which is perpendicular to the first.

one, with one indicating the perfect reconstruction of the input signal.

$$R^2 = 1 - \frac{\|X - W \cdot C\|_F^2}{\|X - \overline{X}\|_F^2} \tag{1.1.5}$$

In the following of this thesis, when a source signal can be reconstructed with fewer synergy components and has higher R^2, the corresponding source signal is said to be more synergetic, or with a higher synergy level.

1.1.2 Joint redundancy

By definition, the degree of redundancy (DOR) is the resources that a system possesses to accomplish a given task. The joint redundancy can be affected by various factors such as the number of joints, the kinematic properties and the dynamic properties of the system. A system is called a redundant system when it has excessive DOR to execute a task.

As mentioned previously, one of the factors which can affect the DOR of a system is the difference between the dimensions of the action space (e.g. the number of joints) and the task space. For example, a human arm, as illustrated in Fig. 1.6, is typically redundant as it has seven degree of freedom (DOF) while it accomplishes tasks in a three dimensional space.

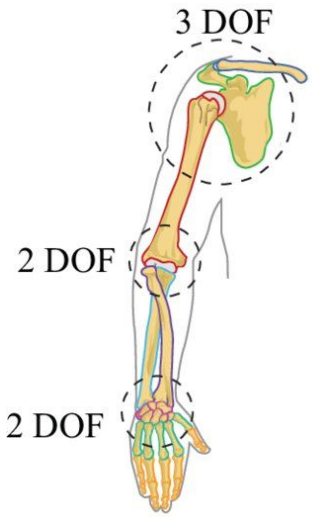


Figure 1.6: A human arm skeleton model with seven degree of freedom. [23]

The kinematic properties of a redundant system, such as its ability to accomplish a task in different ways or motions, are also factors which can affect its DOR. This can be illustrated with the example of an arm trying to reach for an object. If the object is within the range of motion of the arm, there exists many ways to reach for the object, hence it has high joint redundancy in this case. However, if the object is placed far from the arm, there are less possible motions to reach for the object, reducing its joint redundancy in this case.

The dynamic properties of a redundant system,, such as its weight and its force range, can also affect its DOR. For example, a heavily loaded robotic arm can move less freely than an unloaded robotic arm, thus having a lower joint redundancy in this case. The same reasoning can be made with a reduced input-torque robotic arm. All the above factors affecting the joint redundancy are studied in the second chapter of this thesis.

A motivation to quantify the joint redundancy is that it affects the dexterity of a robotic system [20][21][22]. Improved robotic dexterity can equip a robot with various skills to carry out daily tasks, such as picking, sorting, grasping, and manipulating objects. These skills are highly valuable, and the challenging task of creating a dexterous robot is still a topic of active research. Hence, by quantifying the joint redundancy, it can inform users about the suitability of a redundant robot to accomplish a task dexterously.

The joint redundancy analysis in redundant robotic systems is part of the synergy analysis. Indeed, as the synergy is related to the coordination between moving joints, then it is not surprising that the synergy analysis can also review information about the joint redundancy of a robot as demonstrated later in this thesis.

1.1.3 Quadrupedal gaits

A quadruped robot is a bio-inspired robot with four feet that is designed to mimic a quadruped [25]. It is typically studied for the properties of its various gait types. In Fig. 1.7, the different gait types are shown as studied in [24]. The gait diagrams of some quadrupedal gaits are equally shown in Fig. 1.8 [26].

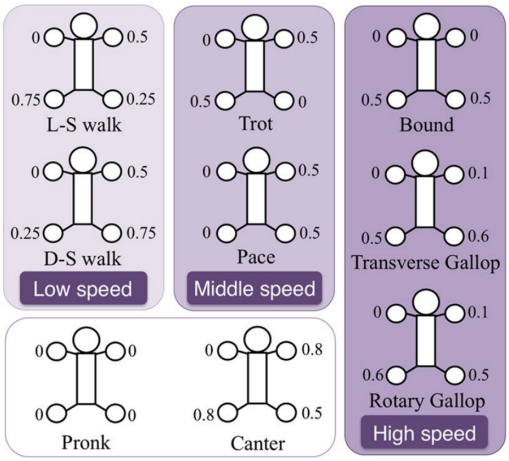


Figure 1.7: Various gait types of a quadruped at moving different speed. The phase delay of each limb is indicated beside the corresponding limb [24].

Studies such as [24][27][28] study the gait types of the quadruped when moving at different speeds and on different terrains. Typically, a quadruped is redundant in the running task which is the task being studied in this thesis. Indeed, a quadruped can be modelled as a multi-joints robot and due to the redundancy of joints, the solution space for the running task is large and it would be hard to control it without specifying beforehand what kind of gait that it will possess in the running motion. This will become one of the motivations for one chapter of this thesis, i.e. to reduce the solution space by specifying the gait type a priori.

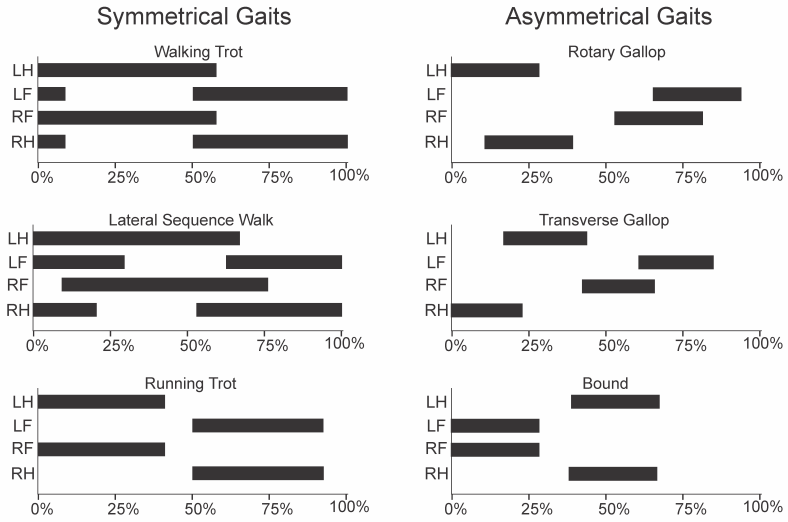


Figure 1.8: Gait diagrams of some well-known quadruped gait types [26]. For each diagram, one cycle of gait is shown, with the black horizontal bars indicates the stance phase of that particular limb. The label LH represents the left hind limb; LF represents the left fore limb; RF represents the right front limb; RH represents the right hind limb.

Besides studying the gait type, in this thesis, it is also desired to demonstrate that different physical properties can affect the energy efficiency of a running quadruped robots, and that a DRL algorithm can exploit the properties to maximize the running performance. To this end, the passive spring-joint effect is studied. When a quadruped is running/galloping at a high speed, it has been shown that there is an efficient interchange between the kinetic energy and the elastic energy stored in the joints, as shown in the Fig. 1.9. The elastic energy is due to the passive joint-spring effects between the joints, and it is common that researchers model the limbs of a quadruped with springs between the joints, as shown in the Fig. 1.10, as the passive joint-spring effect plays an important role in the efficient running gait.

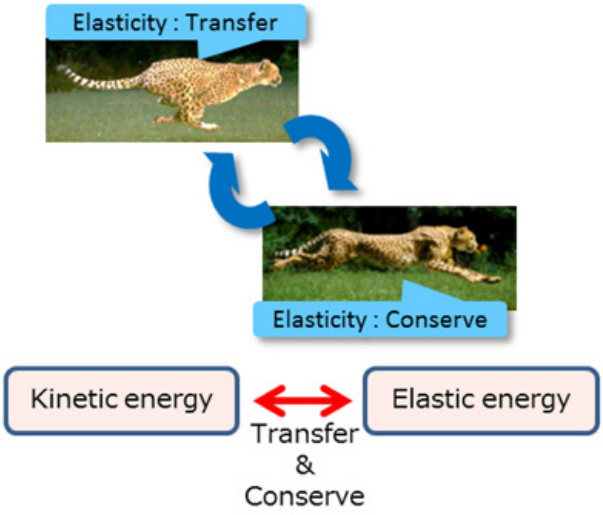


Figure 1.9: The transfer between the kinetic energy and the elastic energy of a cheetah during a galloping motion.

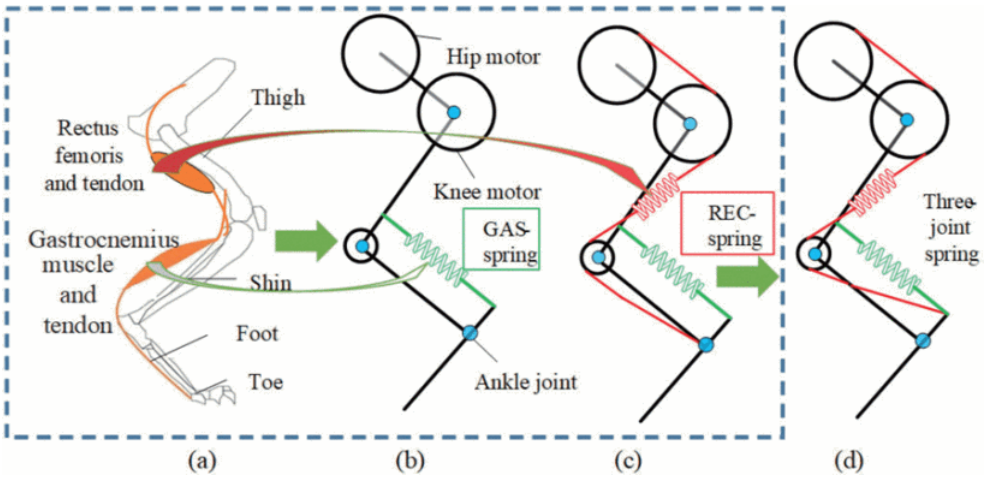


Figure 1.10: The limb model with springs between joints to mimic the passive joint spring effect in a quadruped's limb.

1.1.4 Deep reinforcement learning

Deep reinforcement learning (DRL) is the combination of deep learning techniques and the reinforcement learning theory [29]. DRL is a framework that allows an agent to interact with its environment and adapt its behavior overtime according to the feedback it receives from the environment. DRL has been solving problems that are unsolvable using classical approaches [8][30]. [31] further shows the advantage of DRL over classical controllers in generating energy efficient gaits for snake robots. The potential of DRL becomes the motivation in this thesis to use it as an alternative to classical methods to perform redundant robotic analysis. Indeed, in redundant robotic systems, there are infinite possible solutions to accomplish a task and it is becoming increasingly difficult to use classical methods to control them. DRL is hence the natural choice to search for the optimum solution in the infinite solution space of a redundant system. The necessary background of DRL is explained in this subsection in order to better understand the following of this thesis.

A reinforcement learning (RL) algorithm can be modeled as an infinite-horizon Markov decision process (MDP) which is defined by the tuple (S, A, p, r), where the state space S and the action space A are continuous, and the unknown state transition probability $p : S \times A \times S \rightarrow [0, \infty)$ represents the probability density of the next state $s_{t+1} \in S$ given the current state $s_t \in S$ and action $a_t \in A$, while $r : S \times A \rightarrow \mathbb{R}$ is the reward emitted from the environment on each transition. ρ_π denotes the trajectory distribution induced by a policy $\pi(a_t|s_t)$. Fig. 1.11 illustrates the basic RL framework.

In RL, two quantities are used to characterise the usefulness of a certain state for the agent acting in the environment. The value function $V^\pi(s)$, written as the Eq. 1.1.6, characterises the value of a state by the expected return $\mathbb{E}_{\tau\sim\pi}[R(\tau)]$ that could be obtained from the trajectory τ starting from the state s when the agent acts according to the policy π. Similarly, to quantify the usefulness of an action a when taken in a state s, the Q function, $Q^\pi(s, a)$ is used as shown in the Eq. 1.1.7. The Q function indicates the expected return $\mathbb{E}_{\tau\sim\pi}[R(\tau)]$ that could be obtained from the trajectory τ starting from the state s and action a when the agent acts according to the policy π.

$$V^\pi(s) = \mathbb{E}_{\tau\sim\pi}\left[R(\tau)|s_0 = s\right] \tag{1.1.6}$$

$$Q^{\pi}(s,a) = \mathbb{E}_{\tau\sim\pi}\left[R(\tau)|s_0 = s, a_0 = a\right] \tag{1.1.7}$$

To obtain the true value function $V^*(s)$ and the true Q function $Q^*(s,a)$, the Bellman equations are used to update the value function $V^{\pi}(s)$ and the Q function $Q^{\pi}(s,a)$, which are shown in the Eq. 1.1.8 and the Eq. 1.1.9 respectively.

$$V^{\pi}(s) = \mathbb{E}_{a\sim\pi, s'\sim P}\left[r(s,a) + \gamma V^{\pi}\left(s'\right)\right] \tag{1.1.8}$$

$$Q^{\pi}(s,a) = \mathbb{E}_{s'\sim P}\left[r(s,a) + \gamma \mathop{\mathbb{E}}_{a'\sim\pi}\left[Q^{\pi}\left(s',a'\right)\right]\right] \tag{1.1.9}$$

In DRL, the two functions $V^{\pi}(s)$ and $Q^{\pi}(s,a)$ are often approximated using neural networks with a certain set of parameters, and they can be written as $V_{\psi}(s)$ and $Q_{\phi}(s,a)$, with ψ and ϕ the parameters of the neural networks respectively.

A prerequisite for the studies in this thesis is the choice of DRL algorithms that are capable of solving complex robotic tasks. To this end, SAC [32] and TD3 [33] are chosen as they are the state-of-the-art DRL algorithms. The choice of two algorithms instead of one is to have a less biased choice of algorithms as they are from two different classes of DRL algorithms distinguished by their exploration strategy.

Soft Actor-Critic (SAC) algorithm

SAC is a stochastic DRL algorithm which learns a policy $\pi_{\theta}(a_t|s_t)$ that maps a state of an agent to a probability distribution of actions from which an action is chosen to maximize the objective functions. The particularity of SAC is that it learns a policy by maximizing the expected Q values in parallel with the expected entropy

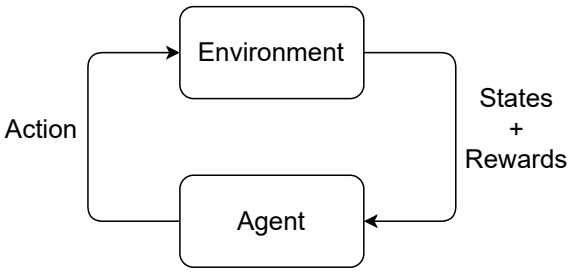


Figure 1.11: The basic reinforcement learning (RL) framework. An agent interacts with the environment by taking some actions. In return, it receives some feedback about the states of the environment as a result of the executed actions. Rewards are also given for each action to enable the agent to adjust its behavior overtime.

of the policy, $H(\pi_\theta(a_t|s_t))$ weighted by a temperature parameter α as shown in the Eq. 1.1.10. The entropy of the policy, $H(\pi_\theta(a_t|s_t))$ can be represented by the Eq. 1.1.11.

$$J_\pi(\theta) = E_{s_t \sim \rho_{\pi_\theta}}[E_{a_t \sim \pi_\theta}[Q_\phi(s_t, a_t) + \alpha \cdot H(\pi_\theta(a_t|s_t))]] \tag{1.1.10}$$

$$H(\pi_\theta(a_t|s_t)) = -log(\pi_\theta(a_t|s_t)) \tag{1.1.11}$$

By maximizing the expected entropy, the authors of [32] pointed out that the learnt policy will have a variety of choices of actions to be taken that bring an equal amount of rewards in a given state. This is shown in their paper to improve exploration and hence speed up learning and reduce greatly sub-optimal solutions.

The cost function for updating the Q_ϕ function is written as the Eq. 1.1.12, where $V_\phi(s_t)$ is the V function represented in the Eq. 1.1.13. ϕ_{targ} denotes the parameters for the target networks.

$$J_Q(\phi) = E_{(s_t, a_t, s_{t+1}) \sim \rho_{\pi_\theta}}[\frac{1}{2}(Q_\phi(s_t, a_t) - (r(s_t, a_t) + \gamma V_{\phi_{\text{targ}}}(s_{t+1})))^2] \tag{1.1.12}$$

$$V_\phi(s_t) = E_{a_t \sim \rho_{\pi_\theta}}[Q_\phi(s_t, a_t) + \alpha \cdot H(\pi_\theta(a_t|s_t))] \tag{1.1.13}$$

Finally, the temperature parameter α is automatically adjusted so that the entropy of the policy, $H(\pi_\theta(a_t|s_t))$ is approximately same as the target entropy, H_{targ} set by the user. The cost function for the temperature α is written as the Eq. 1.1.14.

$$J(\alpha) = E_{a_t \sim \rho_{\pi_\theta}}[-\alpha log(\pi_\theta(a_t|s_t)) - \alpha H_{\text{targ}}] \tag{1.1.14}$$

The detailed training loop of the SAC algorithm is shown in the Algorithm 1.

Twin Delayed Deep Deterministic (TD3) algorithm

TD3 is a deterministic DRL algorithm which learns a policy $\pi_\theta(a_t|s_t)$ that maps a state of an agent to the presumably best action to be taken in this state according to the current policy parameters. The objective to be maximized to learn the policy

is the expected Q values as shown in (1.1.15),

$$J_\pi(\theta) = E_{s_t \sim \rho_{\pi_\theta}}[Q_\phi(s_t, \pi_\theta(a_t|s_t))] \tag{1.1.15}$$

One major drawback of deterministic algorithms is the lack of exploration for new actions during learning which in turn results in local minima. In order to overcome this problem, Gaussian noises are added to the output of the deterministic policy for better exploration.

The cost function for updating the Q_ϕ function in TD3 is written as the Eq. 1.1.16, where ϕ_{targ} and θ_{targ} are the target networks for the Q function and the policy π respectively.

$$J_Q(\phi) = E_{(s_t, a_t, s_{t+1}) \sim \rho_{\pi_\theta}}[(Q_\phi(s_t, a_t) - (r(s_t, a_t) + \gamma Q_{\phi_{\text{targ}}}(s_t, \pi_{\theta_{\text{targ}}}(s_{t+1}) + \epsilon))^2] \tag{1.1.16}$$

The ϵ is a gaussian noise sample introduced by the TD3 authors to improve their algorithm performance. The detailed training loop of the TD3 algorithm is shown in the Algorithm 2.

Algorithm 1 Soft Actor-Critic (SAC)

Initialize policy parameters θ, Q-function parameters ϕ_1, ϕ_2, an empty replay buffer $\mathcal{D}$, the update frequency f, the number updates n, learning rates λ_Q,λ_θ,λ_α
Set target parameters equal to main parameters $\phi_{\text{targ }1} \leftarrow \phi_1, \phi_{\text{targ }2} \leftarrow \phi_2$
repeat
    Observe the current state s and sample an action $a \sim \pi_\theta(\cdot \mid s)$
    Execute a in the environment
    Observe the next state s', the reward r, and the terminal signal d
    Store (s, a, r, s', d) in the replay buffer $\mathcal{D}$
    If s' is terminal (d is true), reset the environment state.
    if iteration *iter* modulo f **then**
        for n times **do**
            Sample a batch of transitions, $B = \{(s, a, r, s', d)\}$ from $\mathcal{D}$
            Update Q-functions by one step of gradient descent using:

$$\phi_i \leftarrow \phi_i - \lambda_Q \nabla_{\phi_i} J_Q(\phi_i) \text{ for } i = 1, 2$$

            Update the policy by one step of gradient ascent using:

$$\theta \leftarrow \theta + \lambda_\theta \nabla_\theta J_\pi(\theta)$$

            Update temperature α with:

$$\alpha \leftarrow \alpha - \lambda_\alpha \nabla_\alpha J(\alpha)$$

            Update target networks with

$$\phi_{\text{targ }i} \leftarrow \rho \phi_{\text{targ }i} + (1 - \rho)\phi_i \quad \text{for } i = 1, 2$$

        end for
    end if
until convergence

Algorithm 2 Twin Delayed Deep Deterministic (TD3)

Initialize policy parameters θ, Q-function parameters ϕ_1, ϕ_2, an empty replay buffer $\mathcal{D}$, the update frequency f
Set target parameters equal to main parameters $\theta_{\text{targ}} \leftarrow \theta, \phi_{\text{targ }_1} \leftarrow \phi_1, \phi_{\text{targ }_2} \leftarrow \phi_2$
repeat
    Select action with exploration noise $a \sim \pi_\theta(s) + \epsilon$, $\epsilon \sim N(0, \sigma)$
    Execute a in the environment
    Observe the next state s' and the reward r
    Store (s, a, r, s') in the replay buffer $\mathcal{D}$

    Sample a batch of transitions, $B = \{(s, a, r, s')\}$ from $\mathcal{D}$
    $\tilde{a} \leftarrow \pi_{\theta_{\text{targ}}}(s') + \epsilon$, $\epsilon \sim clip(N(0, \tilde{\sigma}), -c, c)$
    $y \leftarrow r + \gamma min_{i=1,2} Q_{\phi_{\text{targ}_i}}(s', \tilde{a})$
    Update $\phi_i \leftarrow argmin_{\phi_i} |B|^{-1} \sum (y - Q_{\phi_i}(s, a))^2$
    if iteration *iter* modulo f **then**
        Update θ by the deterministic policy gradient:

$$\nabla_\theta J_\pi(\theta) = |B|^{-1} \sum \nabla_a Q_{\phi_1}(s, a)|_{a=\pi_\theta(s)} \nabla_\theta \pi_\theta(s)$$

        Update target networks:

$$\phi_{\text{targ }_i} \leftarrow \rho \phi_{\text{targ }_i} + (1 - \rho)\phi_i \quad \text{for } i = 1, 2$$

$$\theta_{\text{targ}} \leftarrow \rho \theta_{\text{targ}} + (1 - \rho)\theta$$

    end if
until convergence

1.2 Research Objective

1.2.1 General Objectives

There are three main chapters in this thesis, each having their own objective and are inter-correlated with each other as shown in the Fig. 1.12. The general objectives are:

- Recreate the synergy emergence process and carry out a synergy analysis on simulated redundant agents trained using DRL, before proposing several synergy metrics to evaluate the synergy level of an agent.
- Quantify the joint redundancy of several DRL-trained simulated redundant agents using the results in the first study.
- Propose gait mode specification method to impose a certain synergy mode on simulated quadruped agents trained using DRL, with the aim to carry out an energetic analysis subsequently.

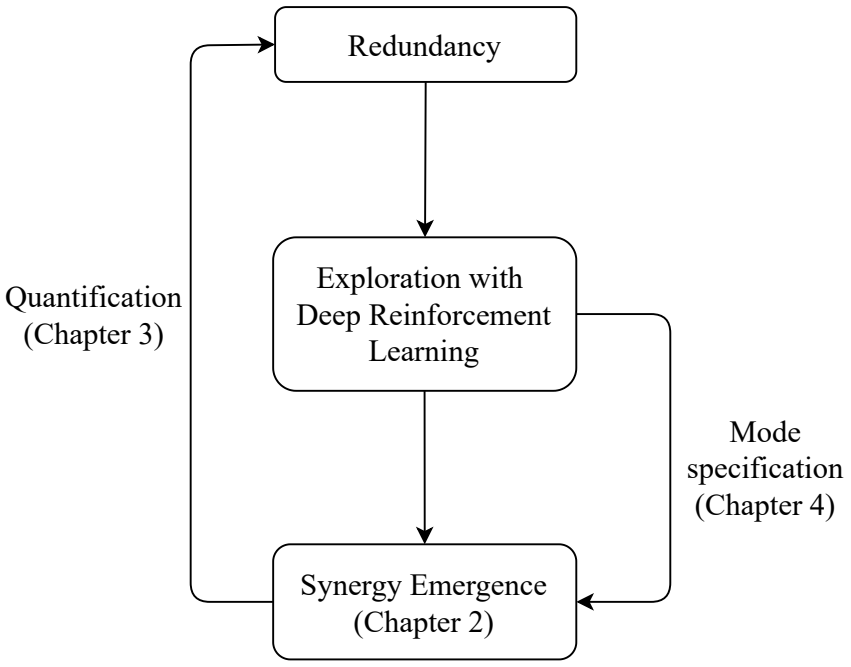


Figure 1.12: The thesis structure overview.

1.2.2 Specific Objectives

- Study the motor synergy concepts and find related works about synergy.

- Adapt the motor synergy concept to the robotic settings where there are only joints in a redundant robot and not muscles as in the case of human.
- Demonstrate the synergy emergence phenomenon.
- Study the relationship between the synergy level, i.e. the level of coordination between joints of a robotic agent, and the performance of the agent for the given task.
- Study the energy efficiency of a trained robotic agent and its association with its synergy level.
- Proposition of several synergy-related metrics to evaluate a DRL algorithm.
- Use one of the previously proposed synergy metrics to quantify the joint redundancy of DRL-trained agents.
- Demonstrate that the proposed redundancy quantification methodology can effectively quantify the relative joint redundancy of several agents executing different tasks with varying number of joints.
- Show that the proposed methodology can also bring extra information, such as the importance of a joint in a certain task.
- Demonstrate that the proposed methodology can also quantify the kinematic and dynamic factors which affect the joint redundancy of a human-like robotic arm.
- Demonstrate the possibility of using the proposed methodology to evaluate the optimality of a robotic structure for a given task.
- As the final part of the thesis, implement a DRL algorithm which can specify the gait mode of a quadruped agent, which indirectly impose a certain synergy mode on the agent.
- Demonstrate that the gait mode specification speeds up the learning process and allows energetic study between two gait modes, i.e. the gallop and trot gaits, for two different forward speeds.
- Demonstrate the advantage of DRL in exploiting the body condition of the robots, i.e. the passive joint-spring effect, similar to the adaptation skills of an animal.

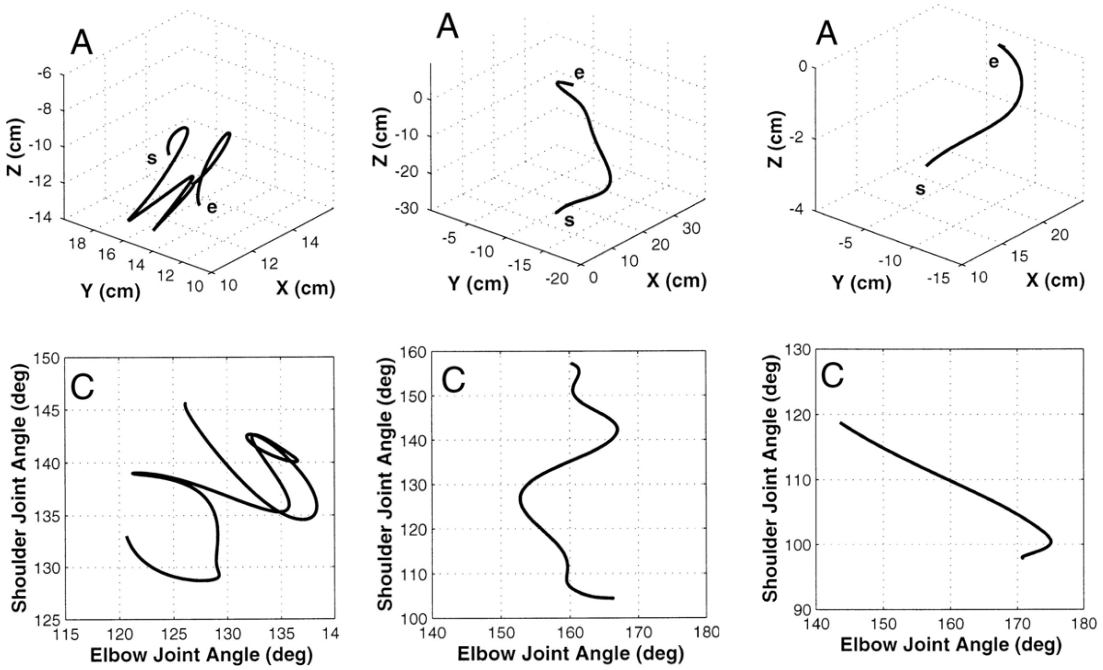
1.3 Related Work

With the background of several key concepts explained in the previous section, works that are closely related to this thesis are presented in this section.

1.3.1 Human Motor Synergy Studies

As the motor synergy concept is originated from the human studies, a few related works on the motor synergy in human motor control are elaborated here. The authors in [6] measured the hand paths of infants at different ages when reaching and grabbing toys, which indirectly revealed the motor synergy development through growth and learning. Fig. 1.13 illustrates the typical movements of infants at the age of 19, 29, and 42 weeks during the pre-reaching, early reaching and stable reaching periods. As illustrated by the rather random motion shown in Fig. 1.13 (a)(A), in the first few months of life, infants follows an indirect route to the target. As they reach and grab for toys faster and more smoothly with age, the high variability of early reaches reduces, as reflected in Fig. 1.13 (c)(A). Also, as shown in Fig. 1.13 (a)(C), the phase portrait between the shoulder joint angle and the elbow joint angle of early reaches clearly shows that the coordination between the shoulder and elbow is poor and variable. However, when it shifts to a stable reach period, the phase portrait shows an aligned combination between the shoulder joint angle and the elbow joint angle, as depicted in Fig. 1.13 (c)(C), which means that the shoulder and the elbow motion become tightly coupled. The result suggests that infants can improve their motor control ability through repetitive practice as they acquire motor synergies for efficient motion.

In [7], the authors described the spatiotemporal organization of the muscle patterns for fast-reaching movements, which is another human motor synergy study related to this thesis. Specifically, they recorded electromyographic (EMG) activity for up to 19 shoulder and arm muscles during point-to-point movements between a central location and one of eight peripheral locations, which were arranged on a circle, either in the sagittal or frontal plane as shown in Fig. 1.14. These movements were performed with different loads on the hand (the experiment 1, see Fig. 1.14 (a), (b)) or with different postures of the forearm (the experiment 2, see Fig. 1.14 (d). They also studied more complex reaching movements (the experiment 3, see Fig. 1.14 (f), (g)): continuous movements from one central or peripheral position to a second position and back to the first (reversal movements), and movements from one peripheral position to a second peripheral position through the central position (warp point movements). The authors tested whether synergies could reconstruct



(a) Pre-reaching period. (b) Early-reaching period. (c) Stable reaching period.

Figure 1.13: Experimental results in [6]. The trajectories of the hand from the start to the end point (A) and the phase portraits between the shoulder joint angle and the elbow joint angle (C) for infants of the age of 19 (a), 29 (b), 42 (c) weeks.

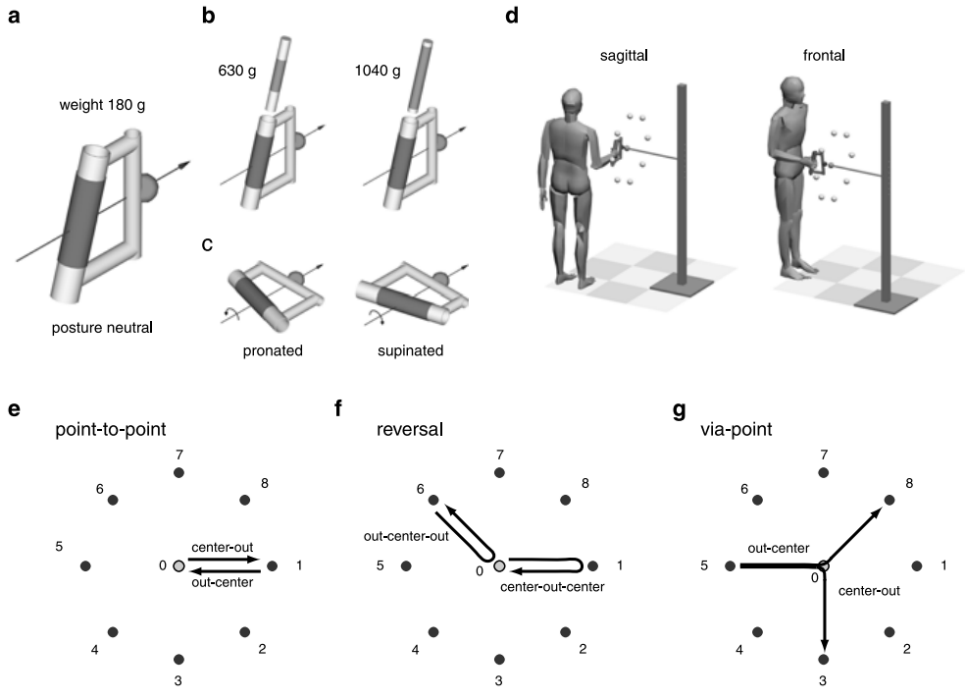


Figure 1.14: Experimental apparatus and conditions in [7] for the control of fast-reaching movements.

muscle patterns of the point-to-point movements. They have demonstrated that the complex characteristics of the muscle patterns for reaching were captured by a small number of combinations of components, suggesting that the mechanism of motor control exploits low dimensionality to simplify control. This is exactly the manifestation of the motor synergies in the human motion when executing the tasks.

In [3][4], the authors studied the synergy development in human learning processes using the PCA. Fig. 1.15 shows the synergy analysis being done in [4]. The discussions in [5] about the role of muscle synergies in simplifying motion generation are also closely related to this thesis as well. On the other hand, the relationship between the muscle synergies, performance, and energy consumption was studied in [34][35], supporting the analysis and results in this thesis. While simulated robotic agents instead of human subjects are studied in this thesis, their concepts of study could be found in this thesis and they justify the study approach used here.

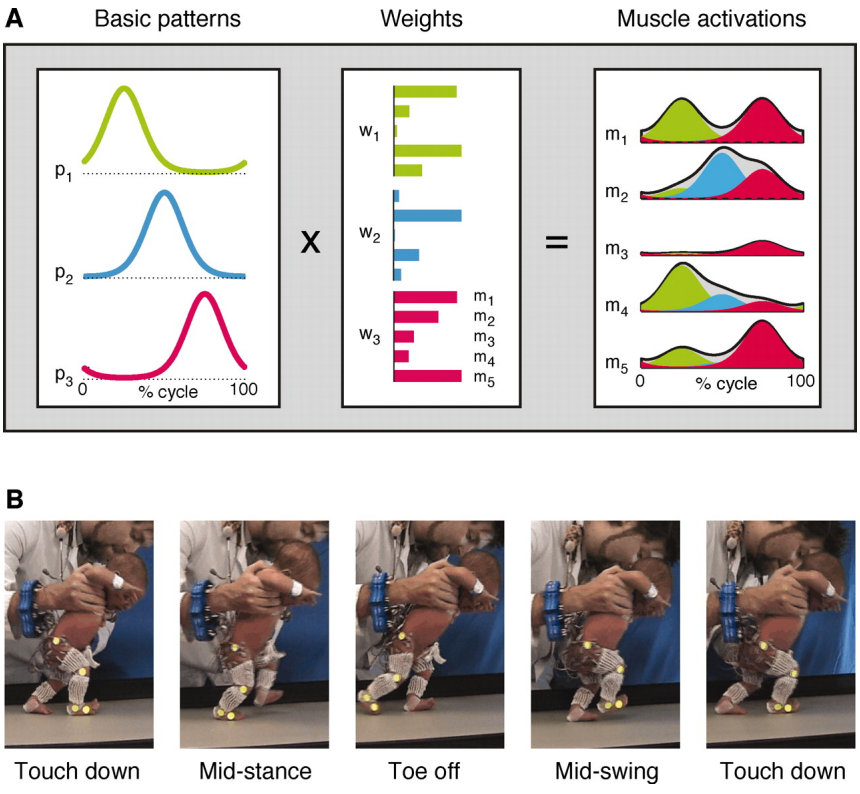


Figure 1.15: The synergy analysis of the human waking as done in [4]. For (A), the weights W_i are the spatial synergy components and the basic patterns are the activation signals C_i as explained in the previous sections. (B) shows the different walking phases of a baby.

1.3.2 Motor Synergy in Robotic Control and Deep Reinforcement Learning

While the motor synergy concept is primarily studied in human subjects, it has been shown in several studies that the motor synergy concept can equally be found and/or applied in the optimal robotics control, and the motor synergy is necessary for the coordination of multiple robotic joints to achieve optimal task performance. In [36], the authors conducted experiments with a simple two-joint simulated arm (Fig. 1.16 A) to study the synergies arising from the optimal motor behavior. The iLQR algorithm is used to calculate the optimal control signals minimizing the trajectory cost functions for the arm. The synergies are then extracted by using a dimension reduction algorithm on the control signals. As shown in the Fig. 1.16 B, the authors found that a small number of synergies are enough to reconstruct the original signal with minimal errors, indicating that optimal movements can be planned in a low-dimensional space.

The authors of [37] aimed to study the role of synergies for robot motor coordination, and used a weightlifting task (Fig. 1.17 A) to demonstrate how synergies can emerge from a trial-and-error learning (claimed to be similar to the reinforcement learning). The authors used two PD controllers with a total of 24 parameters optimized through a random search to control the weightlifter. As shown in Fig. 1.17 B, it has been shown that as the learning progresses, the joint coupling increases with or without a payload, indicating the emergence of synergies. The effort to accomplish the task also decreases overtime for the no payload case, demonstrating that the joint coupling effect exploits the dynamics for energy efficient solution.

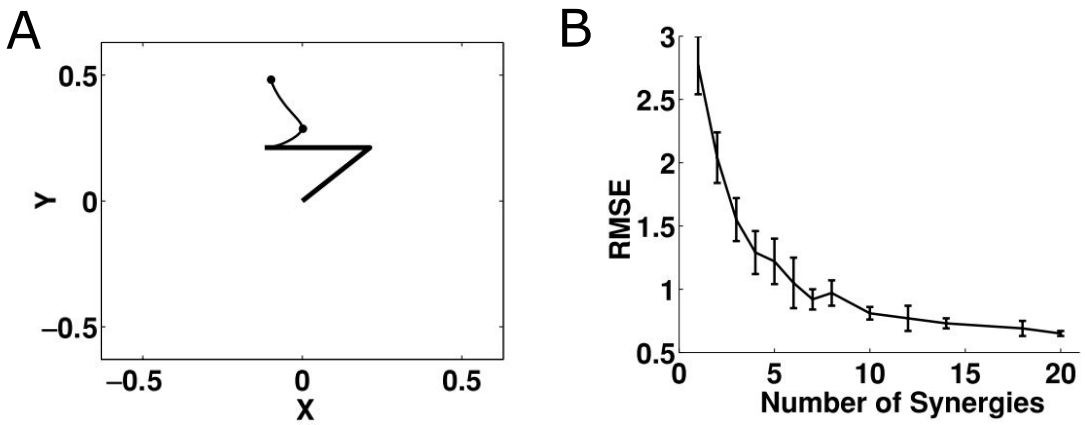


Figure 1.16: The experiment as done in [36]. (A) Via-point movements computed using the iLQR of a two-joint simulated arm. (B) The root mean squared error (RMSE) between actual and reconstructed control signals for via-point tasks as a function of the number of synergies used in the reconstruction. The error bars give the standard deviation of the mean.

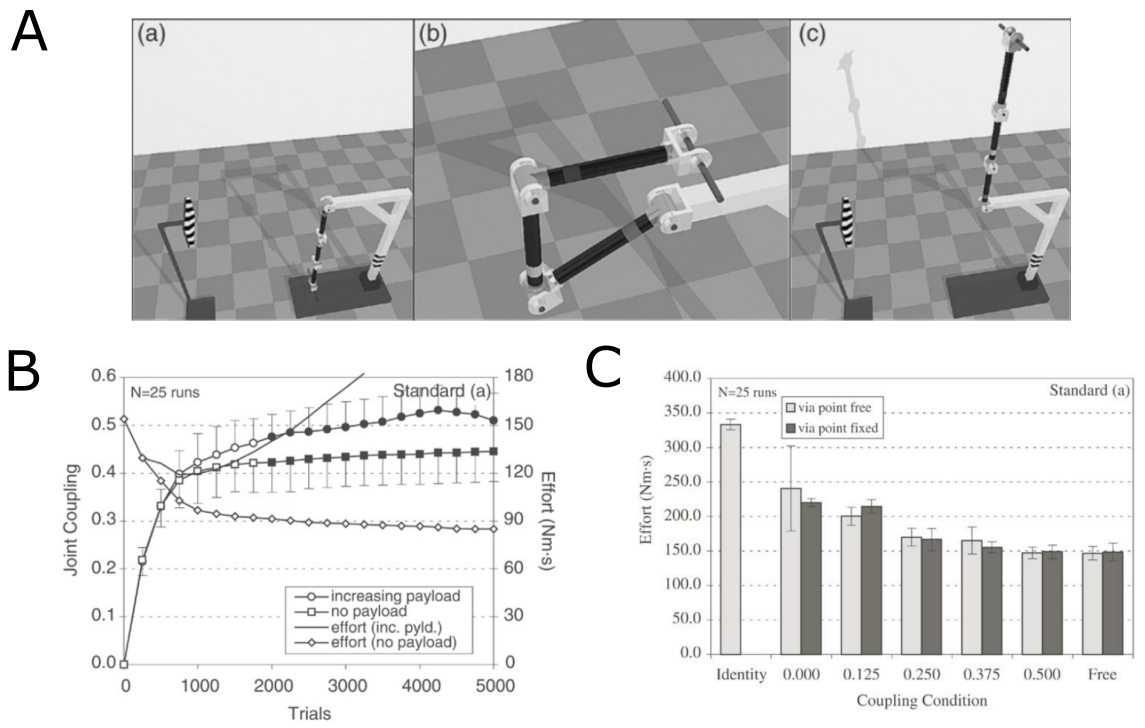


Figure 1.17: The experiment in [37]. (A) The start, via-point, and goal phases of a simulated three-link robotic arm with an obstacle and no payload. (B) Effects of payload and learning on coupling and effort for the standard solution. Pairs of solid markers denote statistically significant differences at the corresponding trial point. (C) Effects of coupling and via points on effort with a 2kg payload for the standard solution.

In Fig. 1.17 C, the authors have also demonstrated that by explicitly controlling the coupling condition of the two PD controllers, it can also be shown that less effort is needed to lift the weight when the coupling increases. As stated by the authors, the increased coupling indicated a progression from independent PD controllers to a more sophisticated nonlinear controller with substantial communication among components for optimal performance and dynamics exploitation. The results of this study are coherent with the human motor synergy studies, suggesting that the synergy concept plays an important role in the robotic control as well.

To the best of our knowledge, there is no known attempt to apply the synergy development concept in DRL algorithms. However, there are a few ideas that have been proposed in classical RL to exploit the joints redundancy of robots by using synergy-inspired strategies. Typically, the authors improve the efficiency of RL algorithms for robotics by assuming that there should be a certain degree of coordination between closely related joints. These joints could be controlled in a low-dimensional latent space with fewer variables than the number of joints and there are a few approaches to find this latent space. The authors of [38] used dimensionality reduction techniques while Expectation-Maximization (EM) approach is used in [39] to

search for this latent space respectively. The authors of [40] further allowed prior structural knowledge about locality synergies to be included in their algorithm by specifying distinct groups of correlated sub-components. For example, all the joints in the arm of a robot can be grouped together by the user to specify that these joints are closely related to each other as a constraint. While the concept of synergy is not used in [41], their analysis on the performance and energy efficiency of RL algorithms is related to the study in this thesis. The authors of [42] also mentioned the energy efficiency issue of RL algorithms in their survey.

As far as we are concerned, there is no closely related work on analyzing existing DRL algorithms throughout the training phase for revealing the relationship between the synergy development, the performance and the energy efficiency. Indeed, the authors of [38][39][40] made the presumption that the joint synergy is indeed the latent variable to be controlled to improve the performance, and accelerate the learning process of RL algorithms. Nevertheless, it is not straightforward about how the joint synergy level develops along with the performance during the learning process. In the first study of this thesis, instead of making the same assumption, a study on the role of synergy in state-of-the-art DRL algorithms for robotics is carried out, conceptually similar to [36][37] but with more complicated simulated robots and new approaches.

1.3.3 Quantification of Joint Redundancy

While a considerable number of studies have focused on the subject of controlling redundant robots [43][44][45][46][47], there have been significantly fewer studies published on the quantification of joint redundancy. A handful of existing studies [48][49][50][51][52][53] have aimed to quantify the redundancy of a robotics joint using model-based approaches, which are one of the most commonly used methods for this subject. There are two notable drawbacks to the use of the classical model-based approach. First, this approach relies on the availability of a mathematical model of the robot, which is usually obtained by making assumptions and simplifications of the actual robotics design to ensure the analytical model is mathematically tractable. These assumptions may not be ideal and the accuracy of the study could be compromised. The second drawback, which is a consequence of the first, is the constraint on the complexity of the robotic structure that can be studied. Indeed, for non-conventional complex robots, this classical approach fails, as the mathematical model will become intractable [54]. The existing studies [48][49][50][51][52][53] were thus limited to basing their studies on canonical redundant robots whose mathematical models are tractable. This poses a problem for modern robotics, as robots

are becoming increasingly complex. Thus, a more scalable method is needed for the quantification of joint redundancy.

Based on the contribution of the first study in this thesis [55] regarding the finding of synergy development process during DRL, one metric in particular, the absolute surface area (ASA) of the synergy development is capable of quantifying the amount of exploration performed by a DRL algorithm before arriving at an optimal control policy to accomplish a given task. When the algorithm needs to explore excessive suboptimal sequences of joint movements during the training phase before finding an optimal solution, this typically indicates that the robotic agent has a high degree of redundancy, and vice versa. Indeed, the excessive exploration required before accomplishing the primary task is a manifestation of high redundancy, as by definition, a redundant robot is one that possesses more resources than those strictly required to execute its primary task [56].

Applying the above concept in the second study of this thesis, the synergy exploration area (SEA) metric, which is renamed after the ASA metric, is used to quantify the relative joint redundancy of a robotic agent between different joint configurations, tasks and dynamical properties. As far as we are concerned, while there are a few works studying the kinematic factors [57][58] which affect the joint redundancy in the robotic field, there is no known work which tries to quantify the kinematic and dynamic factors which affect the joint redundancy.

1.3.4 Quadrupedal Gait Generation and Analysis

Quadruped robot is a common research area and there are numerous research topics which revolve around it, such as energetic studies [27][28][59], design principles [25], gait transition studies [24], etc. These studies play an important role in shedding light on the gait nature of quadrupeds under different circumstances such as varying walking speed [24] and terrain conditions [60], giving us a better understanding of quadrupeds as well as insights on better control strategies for quadrupeds. However, to carry out gait studies on quadruped robots, researchers have always been relying on hand-crafted controllers to generate various gait locomotion on a case-by-case basis [60][61][62][24][59], requiring domain expertise and time-consuming parameters tuning. Therefore, it is desirable to have a more general strategy which allows specific gait generations on quadruped robots with minimum fine-tuning in the search for optimal parameters.

In recent years, deep reinforcement learning (DRL) has been gaining attention as an alternative to classical controllers in quadruped robotic research [10][11][63].

As an example, Fig. 1.18 shows a quadruped being controlled by a DRL algorithm for gait planning as studied in [11]. The use of DRL as an alternative can be due to several advantages that DRL has over classical control strategies. One of the advantages is that the robotic agents trained with DRL has the ability to generalize over various situations unseen during training [64][65], giving the robots adaptation skills similar to animals. DRL also requires less parameter tuning, providing that the reward function is well designed, as the learning process will find a set of optimal parameters via the optimization process.

Unfortunately, DRL has some disadvantages. It is well known that DRL requires long training time [30] and it increases with the complexity of the robots. In the case of quadruped research, there is no guarantee that the DRL-trained agents will finally possess a well-known gait type, making it difficult to carry out the same analysis as in the case of classical controllers. There is however a handful of work such as [66] which imposes a gait type or mode on a quadruped system by introducing prior knowledge in the DRL learning process.

Motivated by the potential of DRL in quadruped system studies, it is desired that some of the downsides of DRL can be overcome by imposing the gait modes, such that it is possible to be used in quadruped energetic studies as presented in the third study of this thesis.

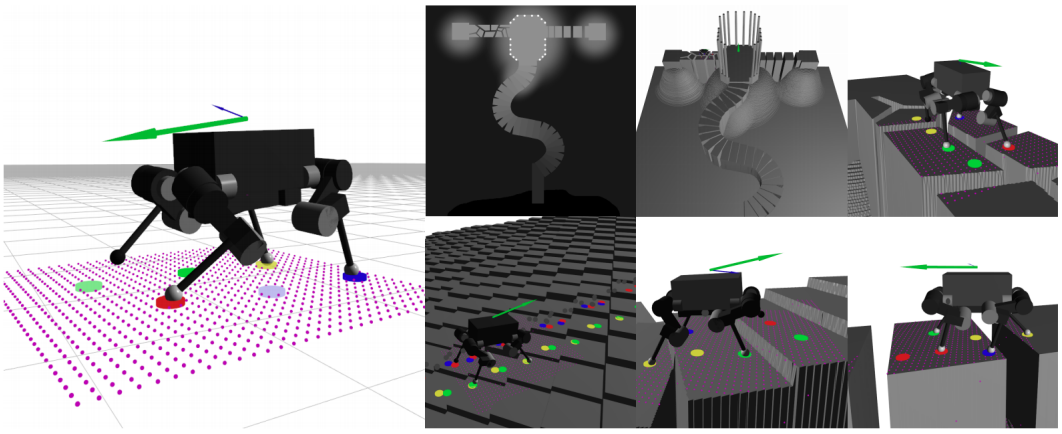


Figure 1.18: The authors of [11] use a DRL algorithm for the gait planning of a simulated quadruped. It is shown that DRL generated gaits are more robust in various terrains.

1.4 Outline

This thesis is organized into five chapters as follows.

Chapter 1 briefly mentions the motivation of this thesis and then the background of some key concepts used in this thesis are explained. Related works of each study in this thesis are also presented, followed by the general objectives and the specific objectives of this thesis.

Chapter 2 presents the first study in this thesis, which is the recreation of the synergy emergence process in simulated robotic systems using Deep Reinforcement Learning. To study the synergy development of the robotic systems, a new framework was proposed to quantify the synergy level of the robotics systems throughout the learning phase of the DRL algorithms. Several synergy-related metrics were also proposed to study the relationship between the synergy level of a robotic agent and the performance of a given task. The energy efficiency of a trained robotic agent and its association with its synergy level was equally studied. The experimental results indicated that the synergy emergence phenomenon could be observed in DRL algorithms, and the motor synergies were required for energy efficient solutions, similar to the results published in studies such as [36]1.17. The results of the study also demonstrated that it is possible to use the proposed synergy-related metrics to evaluate a DRL algorithm statistically.

The first study of this thesis was presented in a paper entitled "Motor Synergy Development in High-performing Deep Reinforcement Learning algorithms" [55], which was accepted for the IEEE Robotics and Automation Letters (RA-L) and also the 2020 International Conference on Robotics and Automation (ICRA).

Chapter 3 describes the methodology to quantify the joint redundancy in DRL-trained redundant robots using one of the synergy-related metrics proposed in the Chapter 2. While it is the same metric used in the previous study, however, its function changes totally in this study. The observations that led to the use of this metric for the joint redundancy quantification was first explained from the perspective of the exploration properties of DRL algorithms. Then, the proposed redundancy quantification methodology was demonstrated to be able to effectively quantify the relative joint redundancy of several simulated redundant agents executing different tasks with varying number of joints. The advantages of the proposed quantification method were further proved through the extraction of useful extra information such as the importance of a joint in a certain task. It has also been demonstrated that the proposed method could quantify the kinematic and dynamic factors which affect the joint redundancy of a 7-DOF human-like robotic arm. Finally, it was also possible to evaluate the optimality of a robotic structure for a given task using the proposed framework.

The second study of this thesis was presented in a paper entitled "Quantification of Joint Redundancy considering Dynamic Feasibility using Deep Reinforcement Learning", which was accepted for the 2021 International Conference on Robotics and Automation (ICRA).

Chapter 4 focuses on the study of a quadruped robot by using a DRL algorithm. Different from the studies done in the previous two chapters, this chapter proposes the use of the DRL algorithm to manipulate the synergy properties of the quadruped agent, i.e. the gait mode of the agent. A new method to specify a gait mode for the quadruped robot was proposed. The results showed that the specification of gait mode sped up the learning process of the DRL algorithm, as well as the synergy emergence process. The study also demonstrated the possibility to carry out an energetic study of DRL-trained quadruped robots with two gait modes, i.e. the gallop and trot gaits, for two different speeds. The advantage of DRL in exploiting the body condition of the robots were also demonstrated.

The third study of this thesis is submitted to the 43rd Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC) as a paper entitled "Deep Reinforcement Learning with Gait Mode Specification for Quadrupedal Trot-Gallop Energetic Analysis".

Finally, chapter 5 summarizes the findings of the previous chapters. The three studies are centered around the motor synergy concepts and they are compared between them. The role of the DRL in the three studies are also discussed and how it can be used in redundant robotic analysis. To conclude, the main contributions of this thesis and possible future works are presented.

Chapter 2

Synergy Development in Deep Reinforcement Learning

2.1 Introduction

In the neuroscience research field, researchers [3][4][5] have shown that the concept of motor synergy exists in the CNS which uses a much smaller set of variables to control a large group of muscles to generate movements. The co-activations of a set of muscles from a smaller number of neural commands, i.e. motor synergies can reduce the burden of CNS significantly while at the same time imposing a certain level of coordination between joints that are closely related to each other for a certain movement. It has been shown in [4][6] that as humans grow and learn, they acquire motor synergies for optimal motor skills. The motor synergy development may explain the ability of human to perform complex movements naturally and energy-efficiently without thinking too much about the way to perform the movements.

Inspired by the role of the motor synergies in human motor skills, researchers [36][37] have conducted experiments in robotic agents to verify if the motor synergy concept is equally present in the robotic optimal control. The experimental results suggest that during the optimization process in search for optimal control strategy in [36][37], the synergy emergence phenomenon is observed as the robotic agents learn to conduct the given task more energy efficiently. These studies suggest that the motor synergy concept does not only play a crucial role in human motor skills, but also in the robotic optimal control field as well.

While the mentioned robotic studies are promising, the authors [36][37] con-

ducted experiments on simple robotic agents and used relatively straightforward optimization algorithms. However, complex robotic agents which possess a lot of redundancy might be a key for the synergy development process. This is because that with the presence of redundancy, as in the case of human beings, it allows the agent to have the ability to opt for the optimal motor skills among other sub-optimal motor skills, giving rise to the synergy emergence process during the learning phase. Therefore, in this chapter, robotic agents with more joint redundancy are employed in the experiments.

A more complicated robotic agent requires a more advanced learning algorithm, ideally one which shares some similarities with the human learning process. The DRL algorithm is the natural choice in this chapter as it is able to find a near optimal solution in a redundant solution space [8][9], besides having a reward system similar to the human goal-directed learning process [12][13][14][15]. It is desired that the synergy emergence process could be recreated in DRL algorithms as well, and provides some understandings on the following questions: How is the coordination between robotic joints gradually being discovered during the training of DRL algorithms? How does this coordination relate to the performance and energy efficiency of the robots? These questions interrogate if the motor synergy plays an essential role in the DRL algorithms in the search for energy efficient motor skills in robotic agents.

In the first part of this chapter, a joint-space synergy analysis is carried out on multi-joint running agents in simulated environments trained using two state-of-the-art deep reinforcement learning algorithms. The global idea of the first study in this chapter is illustrated in Fig. 2.1. In the second part of this chapter, a similar synergy analysis is done on a 7-DOF arm to compare between a classical controller and a DRL algorithm. This second study shows the learning advantage of a DRL algorithm over a classical controller in the search for a synergetic solution, while at the same time shows that the proposed synergy analysis is applicable also to agents different from the first study.

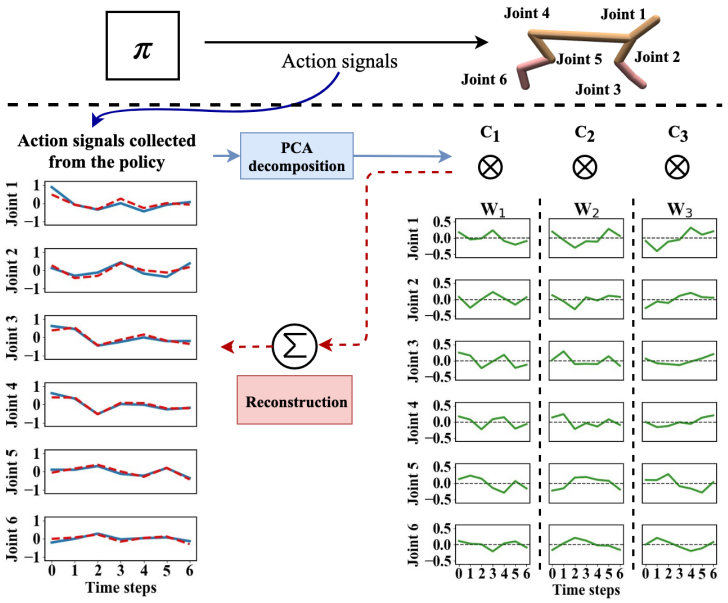


Figure 2.1: Illustration of the joint synergies extraction process using PCA decomposition on action signals (blue) collected from the policy π at certain training checkpoint. The number of joint synergies W to be extracted can be varied during the decomposition. In this example, there are three spatiotemporal synergies represented by the matrices W_1, W_2 and W_3 with the corresponding activation coefficients C_1, C_2 and C_3. The linear combination of W_i and C_i results in a reconstruction of the action signals (red).

2.2 Synergy analysis on DRL-trained running agents

2.2.1 Method

2.2.1.1 Simulated agents

This study is carried out on simulated agents using a simulation engine called MuJoCo [67] that are widely used in studying multi-joint locomotion in the DRL research community. An off-the-shelf robotic agent provided by the OpenAI's Gym library [68], namely HalfCheetah (HC) is used in this study. To show that the study is still valid on the same type of agent with different dynamical properties, a second agent named Heavy HalfCheetah (HeavyHC) is introduced, which has double the weight of HC. For our third agent, the HC is extended into FullCheetah (FC) and the two types of agents are illustrated in Fig. 2.2.

HC and HeavyHC are six-joints agents and FC is a twelve-joints agent which can move in two degrees of freedom (DOF). The motivation behind the choice of these agents is that they are stable in steady-state and do not need any control to maintain an upright position. This property is important as it eliminates unneces-

sary complexities in the joint synergy analysis. Besides, they are also redundant in terms of number of joints for the running task in the forward direction, i.e. a task of two degrees of freedom. The agents are not allowed to deviate from the forward direction, reducing therefore one degree of freedom in the three dimensional space. It would not be easy to design classical controllers to control these two redundant robots for the running tasks as all the joints of the robots need to be coordinated to run forward efficiently. This is also one of the motivations to use DRL algorithm to search for the optimal solution in the vast redundant joint space.

The objective of these agents is to run forward as far as possible in a given duration of time, hence the reward at each time step is the forward velocity of the agent, as shown in the equation (2.2.1). The second term in the reward function (2.2.1), with $A_i(t)$ being the torque input for the joint i, is a penalty by default to impose a little consideration on the energy consumption of the agent during running. The simplicity of this periodic task is suitable for the study of the coordination of the joints as highly synergetic movements are required to move forward quickly and less synergetic control of the joints may lead to inferior performance.

$$R(t) = v(t) - 0.1 \cdot \sum_i A_i(t)^2 \tag{2.2.1}$$

Two reward-related metrics are used to evaluate the quality of the solutions found by DRL algorithms for these agents. The first one is the performance, which measures how far has the agent moved in the forward direction in the given length of time, as shown in the equation (2.2.2) where δt is one time step in the simulation environment. The second metric is the performance-energy index, which is the performance per energy spent during the running task, as shown in the equation (2.2.3) where $E(t)$ is the energy consumption per time step. In other words, the performance-energy index indicates the energy efficiency of an agent. The energy calculation method is discussed in the next subsection.

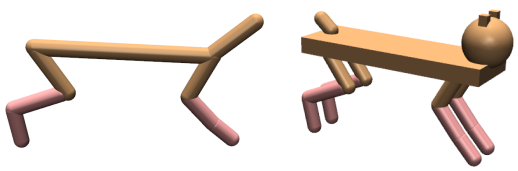


Figure 2.2: HalfCheetah (HC) and FullCheetah (FC) used in our study. Heavy HalfCheetah (HeavyHC) is structurally the same as HC but with its weight doubled.

$$\text{Performance} = \sum_t v(t)\delta t \tag{2.2.2}$$

$$\text{Performance-energy} = \sum_t v(t)\delta t / \sum_t E(t) \tag{2.2.3}$$

2.2.1.2 Energy equation

Energy is one of the key notions used in this study and therefore it is important to clearly define the energy equation used in this study. Studies such as [35][41][69] based their analysis on simulated models and they calculated the energy consumption as a function of the input force to the models. The energy equation at each time step employed in this study is as (2.2.4):

$$E(t) = \sum_i |\tau_i(t) \cdot \dot{\theta}_i(t)| \tag{2.2.4}$$

where i is each joint of the agent, $\tau_i(t)$ being the torque applied and $\dot{\theta}_i(t)$ the angular velocity of joint i at time step t.

2.2.1.3 Joint synergy analysis on simulated agents

In this section, the human motor synergy analysis previously described in section 1.1.1 is adapted for this study. Since the movements of simulated agents are studied here, the control signals generated by the DRL-trained policy will be used as the source signals X in the equation (1.1.3). PCA is then used to solve the equation (1.1.3) for the matrix of joint synergies W and the matrix of activation signals C in the same way as in Fig. 2.1.

Once the joint synergies W and the activation signals C are obtained, the original control signals can be reconstructed with a certain degree of accuracy indicated by the R^2 metric. The R^2 accuracy will vary based on the number of joint synergies used in the reconstruction and this is illustrated on the left in Fig. 2.3. As the number of synergy components increases, the accuracy of reconstruction also increases. The shaded region is the area under the curve calculated by $\int_1^9 R^2 dx$, with R^2 being the accuracy metric, x being the number of synergy components variable. This area quantifies the synergy level of that accuracy curve in the graph.

When the control signals from different training checkpoints of the policy are

collected, the R^2 accuracy curves for these checkpoints can then be plotted on the same diagram and each curve can be associated with different colors from a color gradient, as illustrated on the right of Fig. 2.3. The light green curves correspond to the early phase of training and the purple curves correspond to the ending phase of the training. In this example, it can be noted that the synergy level of the curves increases towards the end of the training, as fewer synergy components are required to achieve higher R^2 accuracy. It is this type of plot, termed synergy development graph, that the analysis in this study is based on as this figure contains rich information about the development of synergetic movements of an agent throughout the training phase indicated by the surface area covered by each curve.

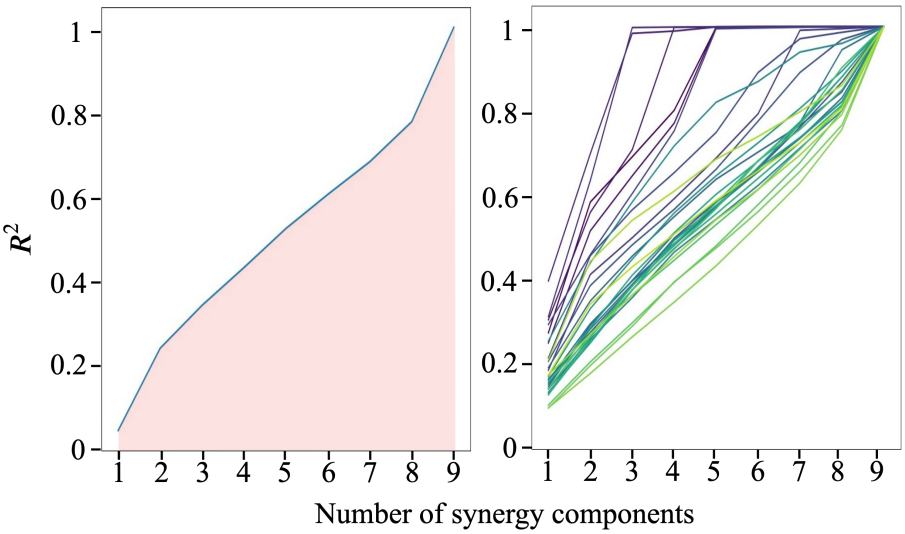


Figure 2.3: Graphs of R^2 accuracy versus the number of synergy components. On the left, the accuracy curve for one training checkpoint is shown, with the shaded region as the area covered by the curve. On the right, the accuracy curves for all checkpoints are plotted on the same diagram with different colors associated with them.

2.2.1.4 Synergy-related metrics

The synergy development graph is useful but it is not convenient as it is in graphical form. Graphical interpretation of experiment results can sometimes be subjective. Besides, if many experiments are carried out, then it will be hard to keep track of all the figures. Hence, it will be better to devise a method to evaluate the synergy development graph quantitatively and objectively. To evaluate the motions of an agent at the end of the training phase, three synergy-related metrics that can summarize the information contained in the synergy development graph are proposed, namely the Final Surface Area (FSA), the Delta Surface Area (DSA) and the Absolute Surface Area (ASA) as illustrated in Fig. 2.4.

The FSA is the area covered by the curve of the last training checkpoint of an agent in the synergy development graph, i.e. the darkest purple line in the graph. This metric could be interpreted as the synergy level of the motions at the end of the training, i.e the compactness of the control policy so that fewer set of synergies are needed to control a larger number of joints. Since human is shown through previous studies [3][5] that synergy is used to achieve a task smoothly, the FSA is expected to be big at the end of the training when the performance of the agent for the given task is high.

The DSA is the area covered between the lines of the last training checkpoint and the first training checkpoint of an agent and could be interpreted as the net improvement in the synergy level of the motions during the learning phase. Generally, the DSA is expected to be positive, i.e. the purple line is above the yellow line. For example, in the human learning case, the synergy level of the gestures of an expert violinist is higher than a novice player in playing a violin [3]. This suggests that during the learning phase of a human violin player, the synergy level of the playing skills also increases. If the DSA is calculated in this case, it should also be positive. A negative DSA indicates a net decrease in the synergy level for the agent's motions.

The ASA is the biggest surface area covered between any two lines in the synergy development graph. This metric can be interpreted as the total action space explored by all the training checkpoints during the learning phase. If different modes of motion are discovered during the learning, this should be visible in the synergy development graph as different motions have different synergy levels. Indeed, by using this metric, the amount of exploration done by any two algorithms during the learning phase for the same neural networks initialization can be quantified and

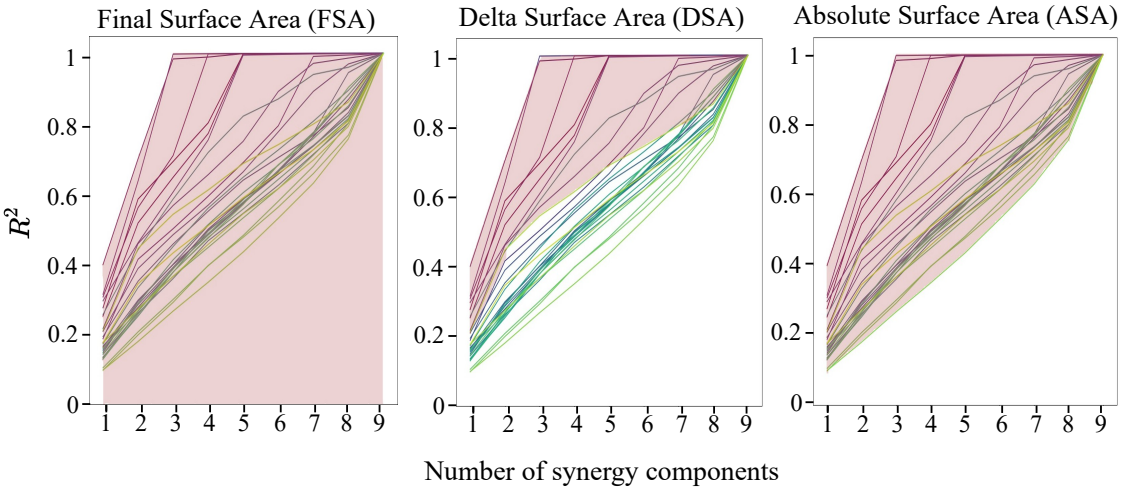


Figure 2.4: The three synergy-related metrics with the shaded regions indicating the area considered in each metric.

compared indirectly. Typically, if the ASA is small, then the agent is likely stuck to a local action space, which can be a good or a bad subspace and further training may not increase the performance significantly. On the other hand, when the ASA is big, the policy is considered to be exploring sufficiently the action space.

While the FSA, DSA and ASA metrics are ad hoc propositions made in this study, they are thoughtfully designed to quantify the results and they provide the required information to draw useful conclusions.

2.2.2 Experimental Results

2.2.2.1 Data collection and preprocessing

For all the experiments, SAC and TD3 algorithms, as described in section 1.1.4, are used to train HC, HeavyHC and FC agents with 15 different random seeds for each algorithm, for a total of 3 million training time steps. The parameters of the algorithms are the same as reported in the original papers [32][33] and the codes are adapted from Softlearning library [70] provided by the SAC authors. During the training, a training checkpoint is kept every 100 thousand time steps. Fig. 2.5 shows the sequence of running motions of HC and FC agents at the end of training.

After the training phase, the actions signals output by the DRL policy for each joint of the agent are collected. This is similar to the human synergy analysis [3][4][5] where the researchers collect electromyography (EMG) signals for the synergy analysis. In this study, the DRL policy analogically plays the role of the CNS, and the action signals output by the policy are analogically similar to the EMG signals. The overview of the data collection and preprocessing process are illustrated in Fig. 2.6. Specifically, in this study, 10 rollouts of 1000 time steps of action signals for

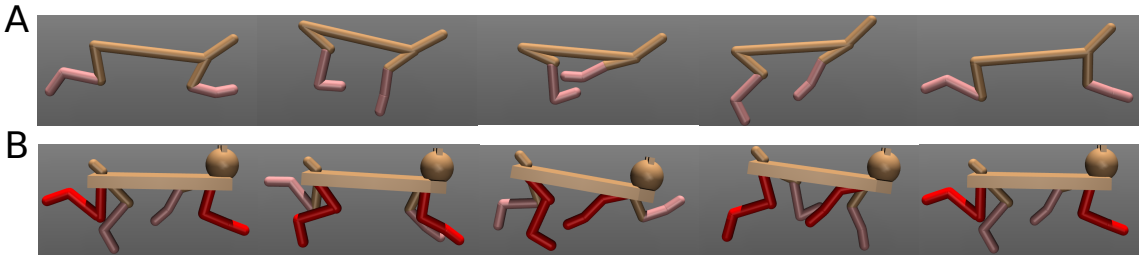


Figure 2.5: The sequence of running motions of (A) HC agent and (B) FC agent after the training phase. For the FC agent, the right limbs are red-colored to better differentiate them from the left side limbs for visualization. At the end of training, the running motions are periodic and the above sequences are representative of the running motions with eventually some variations.

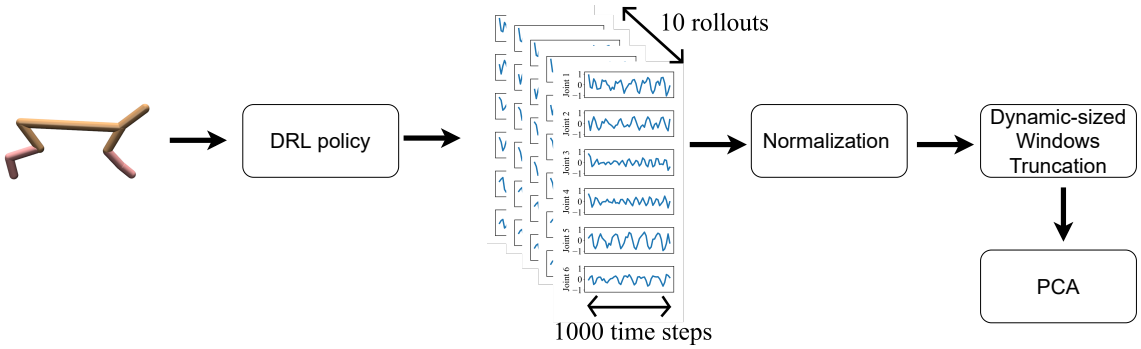


Figure 2.6: The data collection and preprocessing pipeline for one training checkpoint of the DRL policy. The preprocessed data will eventually be decomposed using PCA for the synergy analysis.

the running motion are collected without any exploratory actions for each training checkpoint. As a preprocessing step before extracting the synergies, these collected signals are normalized by subtracting the mean from each dimension of the signals followed by the truncation of each rollout using a time window as the action signals are presumably periodic in a running task. In order to take into consideration the different running speeds of the agent during different checkpoints, adaptive time windows whose sizes change according to the time needed for an agent to run a D distance are used. The adaptive time windows truncation is illustrated in Fig. 2.7. After extracting the joint synergies using PCA, the synergy analysis is carried out as described in the previous section. The energy consumption per rollout is calculated using the energy equation (2.2.4) with 1000 time steps. The source code of this paper can be found at `https://github.com/JiazhengChai/synergyDRL`.

2.2.2.2 Evidence of Synergy Emergence

First, the action signals and the spatiotemporal synergies of one cycle of running motion of a HalfCheetah agent trained using the SAC algorithm are analyzed. The results at the beginning of the training can be found in Fig. 2.8(A) while the results at the end of the training are shown in the Fig. 2.8(B). As it can be remarked in the Fig. 2.8(A) , the action signals between neighboring joints, i.e. joints 1-2-3 and joints 4-5-6, are not correlated which implies a low synergy level as the joints are not coordinated for the running task. The reconstruction with residual errors in Fig. 2.8(A) also suggests that more synergies are needed to represent the action signals. The activation coefficients C_1, C_2 and C_3 are of the same magnitude in this case.

On the contrary, the action signals at the end of the training in Fig. 2.8(B) show clear coordination between joints for the running task, implying a high synergy level. The up-down phase in the signals seems very similar over neighboring joints.

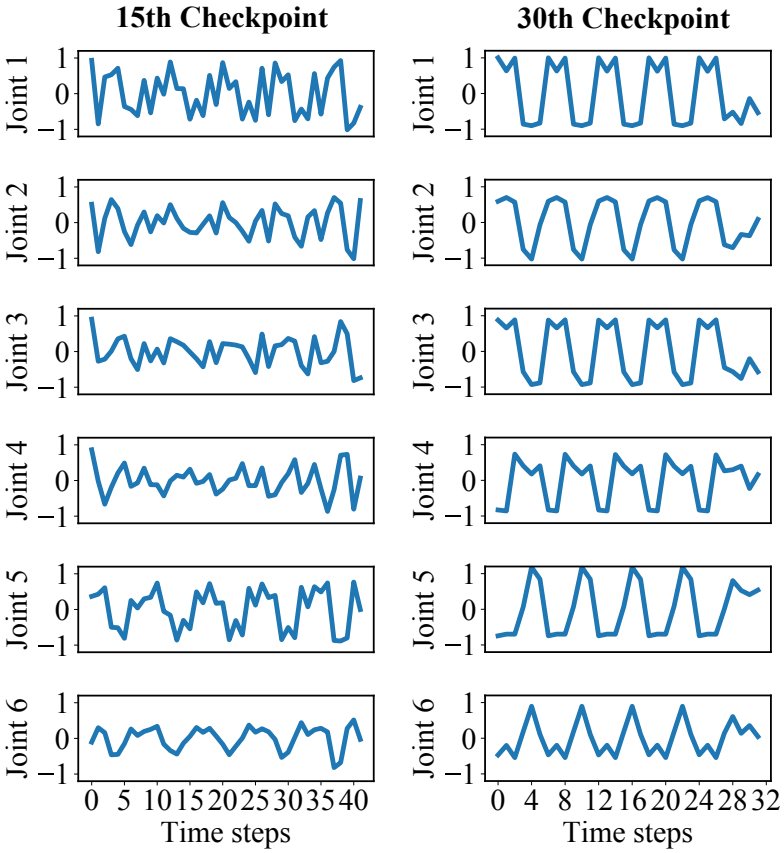


Figure 2.7: The truncated action signals collected at the 15th and 30th training checkpoints of the DRL policy. The truncation window used for the 15th-checkpoint action signals is of 40 time steps in width while the window of the 30th-checkpoint is of 32 time steps in width. This is due to the faster running speed of the 30th-checkpoint HC agent which can finish a predetermined running distance faster than the 15-th checkpoint agent. It can also be noted that the 30th-checkpoint action signals are more periodic and coordinated between joints.

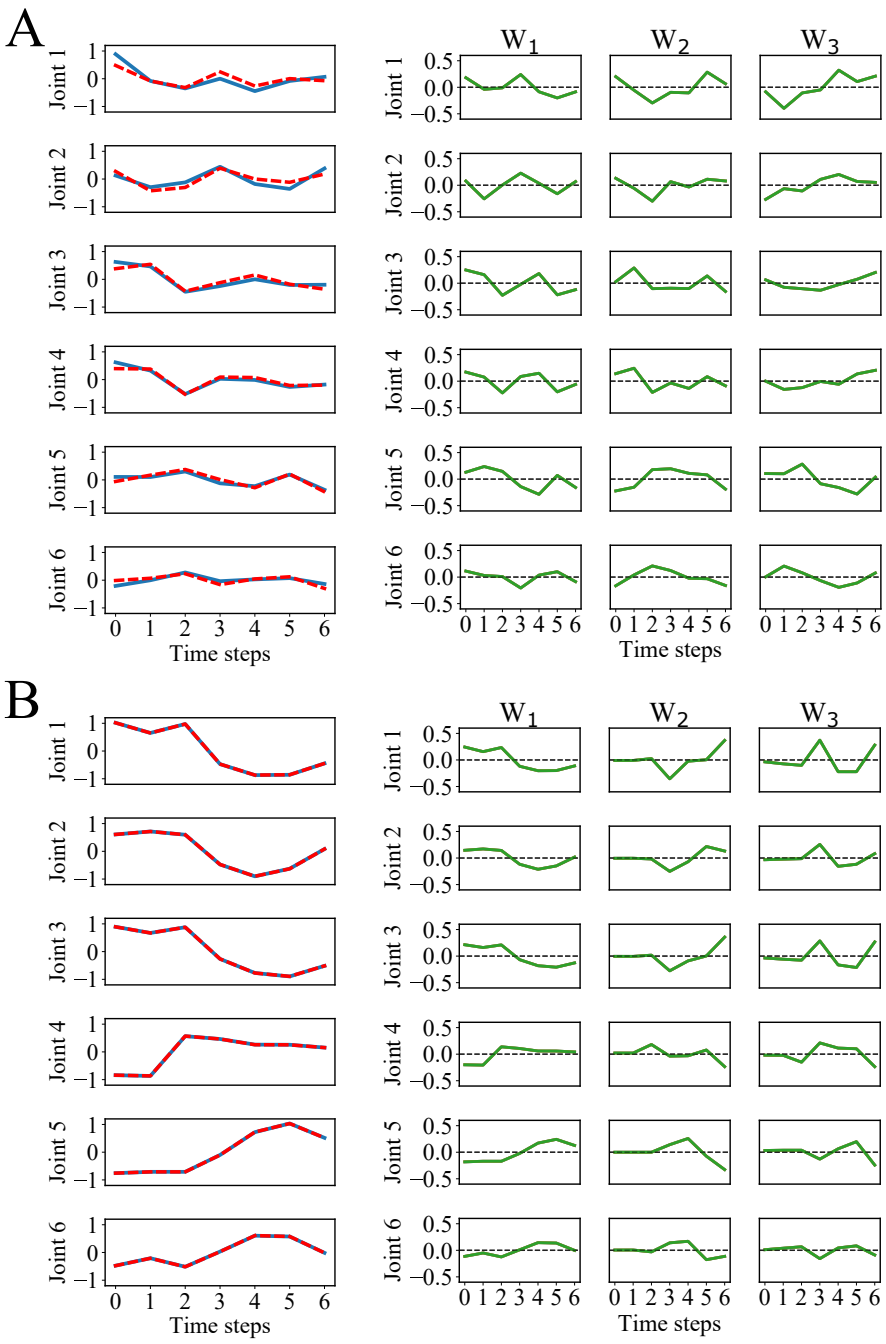


Figure 2.8: (A) The results at the beginning of the training. On the left, the dotted curves (red) are the reconstruction of the action signals (blue) of a HalfCheetah agent. The reconstruction is done by the linear combination of the synergies W_i and the appropriate activation coefficients C_i. On the right, three spatiotemporal synergies of the running motion are shown. (B) The results at the end of the training. The emergence of a common phase over neighboring joints can be remarked.

Indeed, the action signals of the joints 1-2-3 and the joints 4-5-6 are of opposite signs, which suggests that the two limbs of the HalfCheetah agent are moving in opposite directions to create the typical running motion of the front legs and the rear legs of a quadruped. In this case, the activation coefficient C_1 is dominant while C_2 and C_3 are negligible. This suggests that the action signals could be represented with one spatiotemporal synergy at the end of the training for this agent.

The emergence of synergy through the training phase can also be observed using the spatial synergies. In Fig. 2.9, the spatial synergies of the running motion of the HC agent are shown. Before training, the HC agent is not good at using its six joints to execute the running motion smoothly. Indeed, there is no clear pattern of the joint usage before training as shown on the left of the Fig. 2.9. To execute an efficient running motion, it is obvious that the front thigh joint (fthigh) and the back thigh joint (bthigh) must move in opposite phase, i.e. the sign of fthigh joint and the sign of bthigh joint on the left of the Fig. 2.9 must be of opposite signs. This is not the case before the DRL training. However, after the training phase, the emergence of the synergy can be observed and there is a clear pattern of the usage of the joints for executing the running motion as shown on the right of the Fig. 2.9. It can be noted that all the joints of the front limb move in opposite phase with all the joints of the back limb as shown by the opposite signs of the two limbs in the spatial synergy. This emergence of synergy can be observed even it is not explicitly written in the reward function, suggesting that the synergy emergence is part of the obvious solution to run forward efficiently.

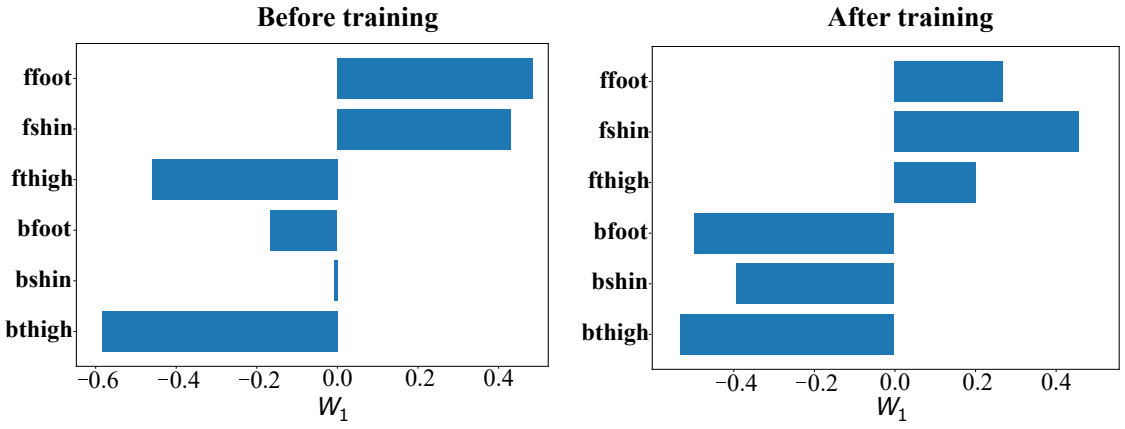


Figure 2.9: The spatial synergies of the HC agent before and after training. Only the most significant spatial synergy component W_1 is shown here. On the vertical axis of both horizontal bar plots, the first letter of each label indicates whether it belongs to the front limb (f) or the back limb (b). The foot, shin, thigh joints correspond to the lower, middle, upper joints of each limb of the HC agent respectively.

2.2.2.3 Synergy Development Graphs

The previous HalfCheetah example demonstrates clearly that as the HC agent being trained using the DRL algorithm, the synergy level increases as the joint coordination increases to execute an energy efficient running motion. Before continuing to evaluate the relationship between the performance, energy efficiency and the synergy level of an agent trained using DRL, it is worth taking a qualitative look at the synergy development graphs of the HC, HCheavy and FC agents throughout the DRL training phase.

Fig. 2.10 shows the synergy development graphs of the three agents trained using SAC and TD3 algorithms. Each row in the figure belongs to an agent as indicated by the title above each plot. Left side plots correspond to the SAC algorithm results and the right side plots correspond to the TD3 algorithm results. The definition of the colors of each curve in all plots is the same as explained in Fig. 2.3, i.e. light green curves correspond to the beginning phase of the training and the purple curves correspond to the ending phase of the training. The more the curves progress towards the upper left corner of the plot as the training progresses, the more synergetic the agent is. This is because that fewer PCA components are required to achieve high R^2 score, indicating that the actions signals for different joints of the agent can be easily represented by fewer variables, showing a high level of coordination between joints for the running task.

For each agent, it can be easily remarked that the SAC trained agents are more synergetic than the TD3 trained agents. The curves of SAC-trained agents progresses towards the upper left region of the plots much more than the TD3-trained agents. Visually, this suggests already that the SAC algorithm is able to find more synergetic running solution than the TD3 algorithm due to its better exploration strategy. It must also be noted that the direction of progression of the curves for the SAC algorithm are generally towards the upper left, while the direction is sometimes not so clear for the TD3 algorithm. Indeed, the curves of the synergy development graphs of TD3 algorithm tend to concentrated around one narrow region and not much progression happens during the training phase. On the contrary, the progression of the curves for the SAC algorithm is obvious and it happens by a large margin. This can again be explained by the exploration strategy of each algorithm, with the TD3 exploration strategy tends to be less effective as it explores only the running solution in a nearby neighbourhood of the current solution, resulting in the narrow synergy development graphs as shown in Fig. 2.10.

In most of the time, the DRL optimization process can find an optimal solution

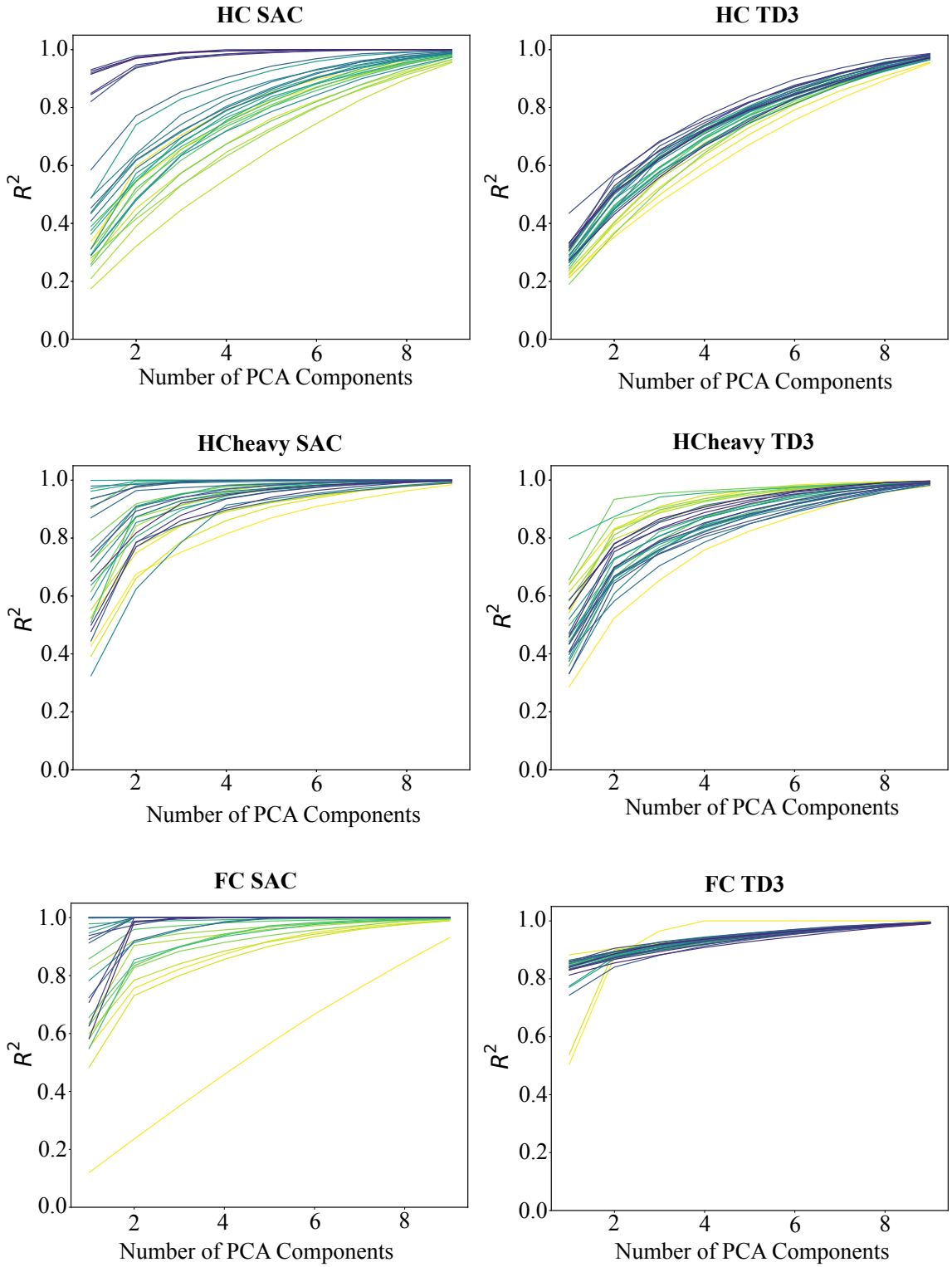


Figure 2.10: The synergy development graphs of HC, HCheavy and FC agents trained using the SAC and the TD3 algorithms. The top row shows the HC synergy development graphs; the middle row shows the HCheavy synergy development graphs; the final row shows the FC synergy development graphs. The SAC and TD3 results of each agent are aligned side by side so that they are easily comparable between them.

and the synergy level will increase throughout the training as shown in most of the plots in Fig. 2.10. However, in some rare cases, the DRL algorithm will fail to find an optimal solution due to reasons such as bad parameters initialization which is normally out of the user's control, the synergy development graph will show a negative progression of the curves throughout the training, i.e. the synergy level of the agent decreases throughout the training, as shown in the Fig. 2.11. In such cases, the DRL algorithm converges to a suboptimal solution and the resulting performance and energy efficiency of the agent are also inferior to the normal cases. This is interesting as the synergy development graph can provide information on the quality of the found solution and this shows potential to be exploited for further applications of the synergy development graph.

While the qualitative reasoning of the synergy development graphs can already give some insightful ideas about the properties of the SAC and TD3 algorithms, it is always better to have a quantitative evaluation of all experiments in order to be able to judge them objectively. Indeed, the synergy development graphs in Fig. 2.10 are only one sample among the 15 experiments that are carried out for each agent. It is impossible to keep track of all the results qualitatively. In the next section, the three

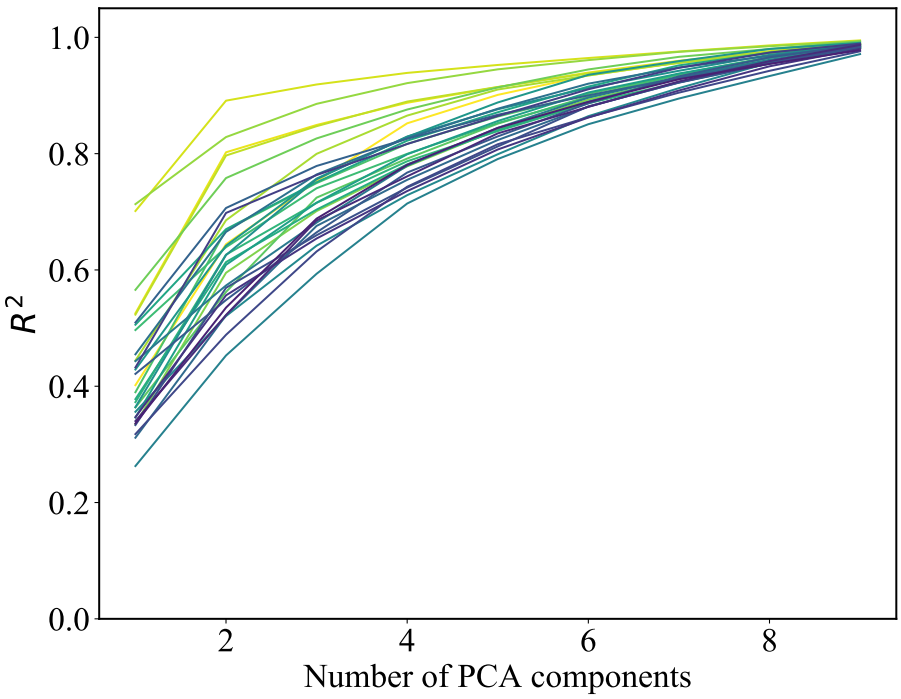


Figure 2.11: The negative progression of the curves, i.e. the synergy level decreases throughout the training, in a synergy development graph when the DRL optimization process fails to find an optimal solution and converged to a suboptimal solution instead.

synergy-related metrics are used to evaluate the experiment results quantitatively.

2.2.2.4 Performance, Energy efficiency and Synergy-related metrics Evaluation

With the concept of synergy level and joint coordination illustrated via the previous HalfCheetah example, a more detail analysis can be done on the relationship between the development of the performance, the performance-energy and the synergy level of the three agents during the training phase where the synergy level is represented by the surface area defined in Fig. 2.3. As shown in Fig. 2.12, the synergy level for the SAC experiments develops more significantly and correlates well with the two reward-related metrics, especially towards the end of the training. On the other hand, the synergy level for the TD3 experiments remains almost constant or even decreasing while the reward-related metrics are gradually increasing during the training phase. If Fig. 2.12 is analyzed carefully regarding the growth of the synergy level represented by the surface area and the growth of the performance-energy index, it can be noted that they are globally correlated. If the growth of the performance-energy index is minimal, the change of the synergy level is also minimal. The higher synergy level and the better reward-related metrics for the SAC algorithm suggest that the synergy might be indeed the latent variable to improve the learning process of a RL algorithm, which has been questioned in the earlier part of this study.

Next, the policies are evaluated at the end of the training phase using the proposed synergy-related metrics. The results of these metrics are presented in Fig. 2.13. To show the pertinence of each metric, the p-values of the Two-Sample t-Tests are calculated between the results of the two algorithms for each agent. If the p-values are smaller than 0.05, it indicates that the difference between the results of SAC experiments and TD3 experiments are significant and that this particular metric can differentiate SAC-trained agents from TD3-trained agents. It can be remarked that the synergy-related metrics, with the exception of the DSA metric, distinguish clearly SAC and TD3. The FSA, which represents the synergy level of the agent at the end of the training, is higher in SAC than TD3. This suggests that SAC tends to find solutions that result in more synergetic motions for the robot agents and this property cannot be observed via the usual performance metric. It should be noted that a high synergy level might suggest that uncomplicated motions are learnt by the agents and can possibly have a negative impact on the performance. However, this is not the case for the results in this study as shown previously in Fig. 2.12.

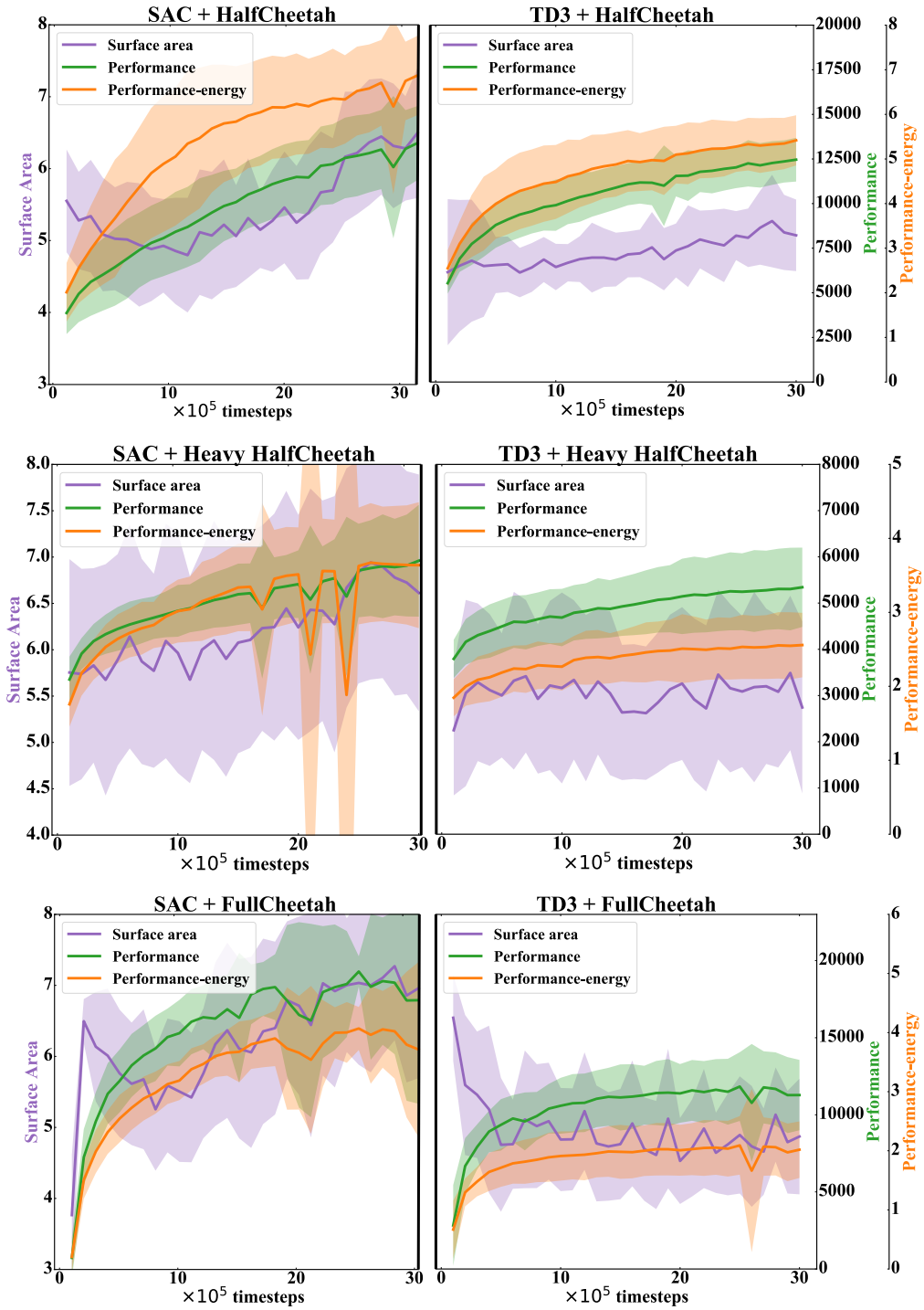


Figure 2.12: Graphs showing the development of the synergy level represented by the surface area (purple), the performance (green) and the performance-energy (orange) for the three agents during the training phase. The three vertical axes of different scales on the same row are common for the two graphs on that row.

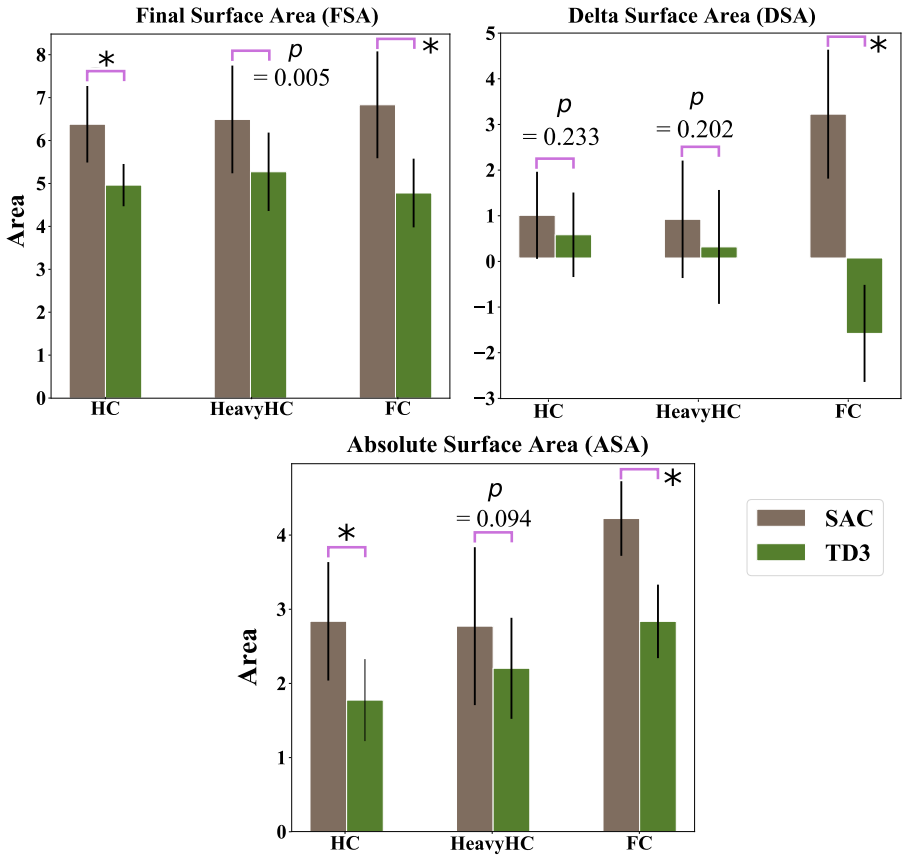


Figure 2.13: Bar plots showing the results of the three synergy-related metrics for three different agents trained with SAC and TD3. The p-values for the Two-Sample t-Tests between each agent are given above the corresponding bars. * is to denote $p < 0.001$.

Secondly, the DSA, which shows the net improvement in the synergy level after the training phase, is also higher in SAC experiments than in TD3 experiments. Although the standard deviation is higher, the DSA of SAC is positive on average but TD3 fails to have this property. This might be related to the exploration strategy of TD3 as it explores by adding noise to the actions suggested by its policy. This random exploration of the action space might cause the synergy level of the motions to change rapidly without any specific order. SAC, on the other hand, explores more systematically by maximizing the entropy of the policy. This enables the policy to propose systematically better actions, which is translated by the positive DSA. The ASA, which represents the action space explored by all training checkpoints, is likewise higher in SAC than TD3. Noting that the neural networks are randomly initialized for both algorithms, this suggests that SAC explores more than TD3, which is proven in SAC paper [32] as a result of their entropy-maximizing exploration strategy. It equally implies that SAC gets stuck to local solution less often, which matches with the higher performance shown in Fig. 2.12.

Now, the reward-related metrics for the two algorithms are compared at the end of the training, which are shown in Fig. 2.14. The performance of SAC is better than TD3, which is already proven in [32]. The performance achieved by TD3-trained agents is still acceptably competitive with SAC in the simpler HC and HeavyHC cases as indicated by the higher p-values of the Two-Sample t-Tests. However, if the two algorithms are examined closer by looking also at the energy consumption of the agent during the task, the energy consumed is lower in SAC than in TD3, resulting in higher performance-energy for SAC. As a result, the performance-energy index shows that SAC is overall better than TD3, achieving a solution that allows the agent to achieve high reward with lower energy consumption.

From these results, it can be deduced that a DRL algorithm cannot be evaluated using only the performance metric. This metric fails to distinguish the SAC and TD3 algorithms while the performance-energy index and the proposed synergy-related metrics indicate that the two algorithms are indeed distinct from each other based on the Two-Sample t-Tests results. The proposed synergy-related metrics show that SAC is one example of a DRL algorithm that has very nice properties in terms of the synergy concept which may explain its better performance.

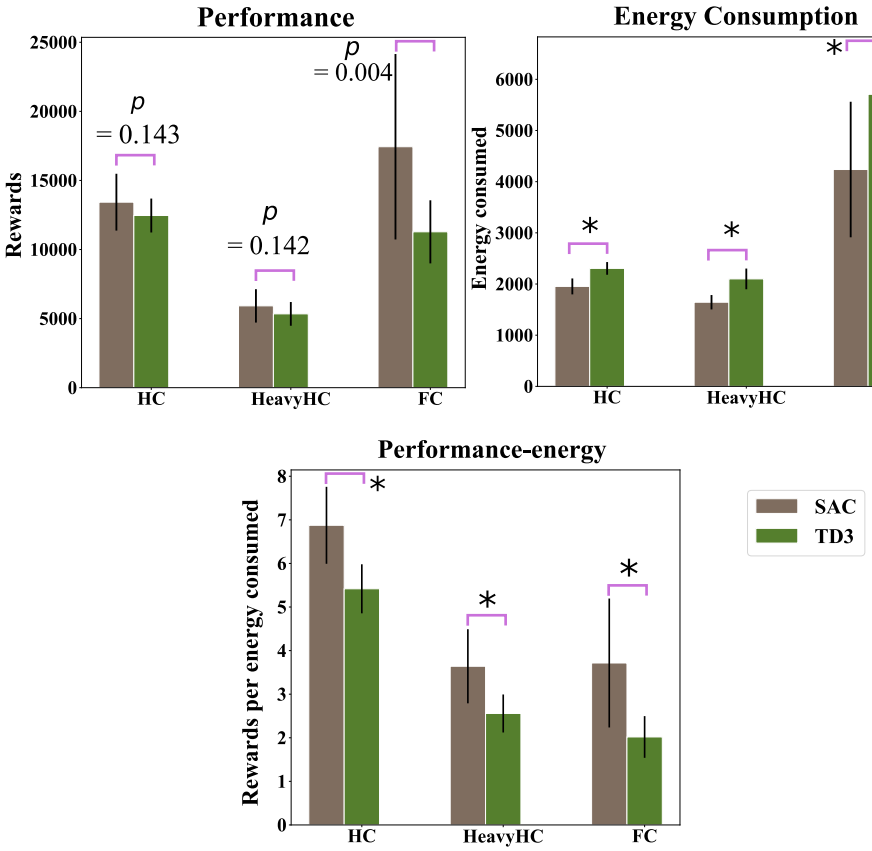


Figure 2.14: Bar plots comparing the results of the usual performance metrics, and also two energy-related metrics between SAC and TD3 for all agents. The p-values are presented above the corresponding bars.

2.2.3 Discussion

For this study, if the reward-energy-related metrics is associated with the synergy-related metrics, there are some interesting implications. First, the higher synergy level in SAC, indicated by the FSA and the DSA, does mean higher performance for the agents. This can be further extended by observing that a higher synergy level also means higher energy efficiency in accomplishing a task. The biological motivation behind this outcome can be found in the work of [34] where the authors did experiments on human subjects and showed that the use of synergies led to higher rowing economy, which is equivalent to the energy efficiency metric in the task of rowing. The similarity with the human study [34] provides convincing supports for the outcomes of this study.

2.3 Synergy analysis on DRL-trained 7-DOF robotic arm

In this second part of the chapter, a similar synergy analysis is done on a 7-DOF robotic arm. However, in this second part, there are two objectives. First, the performance and energy efficiency of the robotic arm controlled by either a classical PD controller or the SAC algorithm are compared. By doing this, it is hoped that the advantages of a DRL algorithm over the classical PD controller could be shown. In particular, the learning ability in the DRL algorithm should make the control strategy more accurate and energy efficient, which is one of the main motivation of this thesis. Second, this second study of the chapter is intended to demonstrate that the synergy analysis done on the running agents in the first part of this chapter is also applicable to a different kind of agent, the 7-DOF robotic arm. It is desirable that the synergy emergence phenomenon could also be observed in the DRL learning phase of the humanoid arm.

2.3.1 Method

2.3.1.1 7-DOF simulated robotic arm

In this study, a 7-DOF robotic arm simulated by the MuJoCo engine is used, as depicted in Fig. 2.15. In the following, this agent will be named as the Arm3D agent. The Arm3D agent comprises three parts, i.e. a 3-DOF shoulder joint for abduction, flexion, and rotation; a 2-DOF elbow joint for flexion and pronation; and

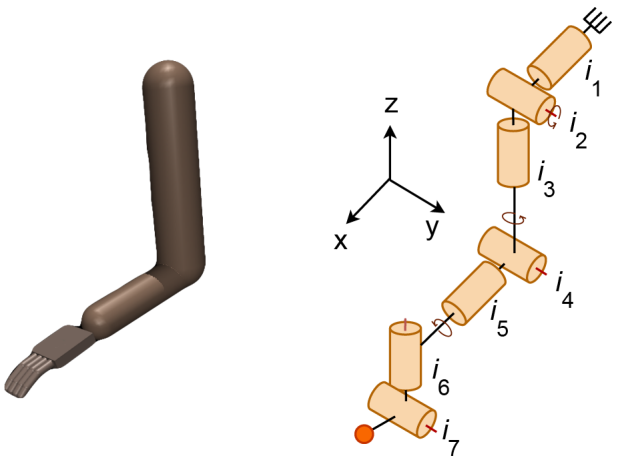


Figure 2.15: The Arm3D agent (left) and its corresponding kinematic diagram (right). Joints i_1 to i_3 enable shoulder abduction, flexion, and rotation, respectively; i_4 and i_5 enable elbow flexion and pronation, respectively; and i_6 and i_7 enable wrist flexion and abduction, respectively. The orange extremity is the fingertip.

a 2-DOF wrist joint for abduction and flexion. Based on the 7-DOF Arm3D robotic agent, a 3-DOF Arm3D agent is also created by allowing only the joint rotations in the sagittal plane, i.e. the joints i_2 , i_4, and i_7 in the Fig. 2.15. The motivation of using two versions of the Arm3D agents in the experiments is to compare the consistency of the results from both the lower DOF and the higher DOF Arm3D agents.

For the physical properties (e.g., body segment dimensions and weights) of the Arm3D agent, it is made sure that the physical properties of the robotic agent are suitable and reasonable that a feasible solution can be found through the DRL algorithms by considering the agent's dynamic conditions. Specifically, the arm segment length and weight are set as the following, with l_1, l_2, l_3 correspond to the length of the upper arm, lower arm, and the wrist respectively, while m_1, m_2, m_3 correspond to the weight of the upper arm, lower arm, and the wrist respectively.

$$\begin{aligned} l_1 = 0.35\,[m] \quad l_2 = 0.23\,[m] \quad l_3 = 0.08\,[m] \\ m_1 = 1.98\,[kg] \quad m_2 = 1.18\,[kg] \quad m_3 = 0.45\,[kg] \end{aligned} \tag{2.3.1}$$

After having described the two variations of the Arm3D agent, two different tasks are assigned to each variation of the Arm3D agent, as shown in the Figure 2.16. For the 3-DOF Arm3D agent on the left of the Figure 2.16, the planar point tracking task is assigned and the agent need to track the moving red target point as accurately as possible in the sagittal plane. To be more precise, the red target point will move in the 2D plane according to the Eq. (2.3.2), where p_1 and p_2 are

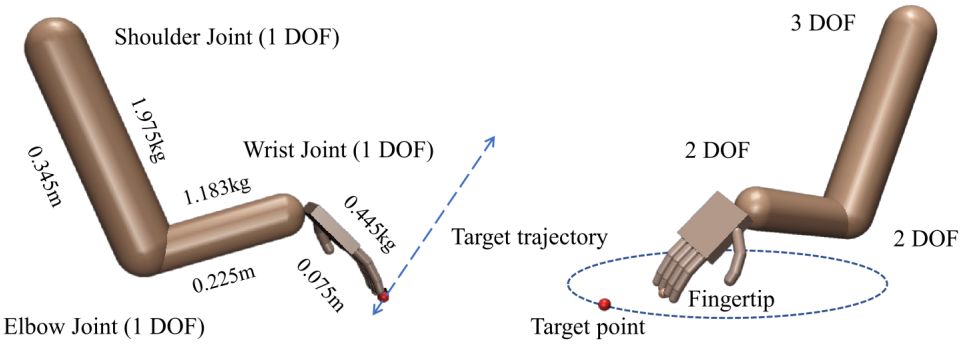


Figure 2.16: The point tracking tasks assigned to the Arm3D agents. On the left of the figure, the 3-DOF Arm3D agent is assigned the planar point tracking task. The red point is the target that moves back and forth along the blue dotted trajectory. On the right of the figure, the 7-DOF Arm3D agent is assigned the circular drawing task in the 3D space.

the two extremities of the straight line fixed manually before the experiments, and f being the frequency of the point.

$$q(t) = (p_1 - p_2)sin(2\pi ft)/2 + (p_1 + p_2)/2 \tag{2.3.2}$$

For the 7-DOF Arm3D agent on the right of the Figure 2.16, the circle drawing task in the 3D space is assigned. The red moving target moves in a circular trajectory according to the Eq. (2.3.3), where p_c is the center of the circular trajectory, and the second term is the vector specifying the circular trajectory in the 3D space. In both the straight line tracking and the circle drawing tasks, the red target is moving with a frequency f of 0.5 Hz.

$$q(t) = p_c + [-0.18\sin(2\pi ft) \quad -0.18\cos(2\pi ft) \quad 0]^T \tag{2.3.3}$$

The choice of the planar point tracking task and the circle drawing task are not by chance. In the motor control domain, the planar point to point reaching is well-known and it can be found in previous studies such as [69][7][71]. The simplicity of this task is important as it allows for easy understanding and analysis of the experimental results. The circle drawing task assigned to the 7-DOF Arm3D agent can be found in studies such as [72][73][74]. It is a more difficult task as it requires the coordination of both shoulder and elbow joint movements to draw the circle smoothly.

Similar to the running-agent study in the first part of this chapter, all dynamics

parameters, such as the segment inertia and the mass, as well as the model of the Arm3D itself, are completely blind to the DRL algorithm, more specifically the SAC algorithm. This study aims to minimize the tracking error between the finger of the Arm3D model and the red target point while considering the energy efficiency of the movement. Hence, the reward function for the SAC algorithm is set as shown in the Eq. (2.3.4):

$$R(t) = -\omega_1 \cdot ||\overrightarrow{fingertip}(x, y, z, t) - \overrightarrow{target}(x, y, z, t)||_2 - \omega_2 \cdot \sum_i A_i(t)^2 \quad (2.3.4)$$

where $||.||_2$ denotes the Euclidean Norm; $\overrightarrow{fingertip}(x, y, z, t)$ is the vector representing the position of the finger tip in the 3D space; $\overrightarrow{target}(x, y, z, t)$ is the vector representing the position of the target point in the 3D space; $A_i(t)$ the input torque at each joint i; ω_1 and ω_2 are the constant coefficients to give different weightings between the first and the second term in the Eq. (2.3.4). The first term is the distance between the target position and the current fingertip position at each time step, while the second term is for the energy expenditure consideration during the movement. ω_1 is set to be always larger than ω_2 so that the tracking error is minimized first before considering any form of energy saving in the optimizaiton process.

The SAC training length for the 3-DOF Arm3D agent are 30 thousand training steps while for the 7-DOF Arm3D agent, 60 thousand training steps are used. During the training process, one roll-out of the arm motion is kept every 1000 training steps. Thus, for all training steps, a total of 30 rollouts are collected for the 3-DOF Arm3D agent while a total of 60 rollouts are collected for the 7-DOF Arm3D agent. The length of one training step is 1000 simulation steps. The training steps were adjusted based on the complexity of the Arm3D agents in order to converge in terms of task performance.

In the current study, an emphasis are put on analyzing the coordinated movements between wrist, shoulder and elbow joint. Hence, the fingers' degree of freedom is not considered.

2.3.1.2 Classical PD controller

The classical PD controller that is used in this study to compare to the SAC algorithm is illustrated in Figure 2.17. It is the similar classical PD controller described in [75] where the authors carried out their study on a robotic model structurally

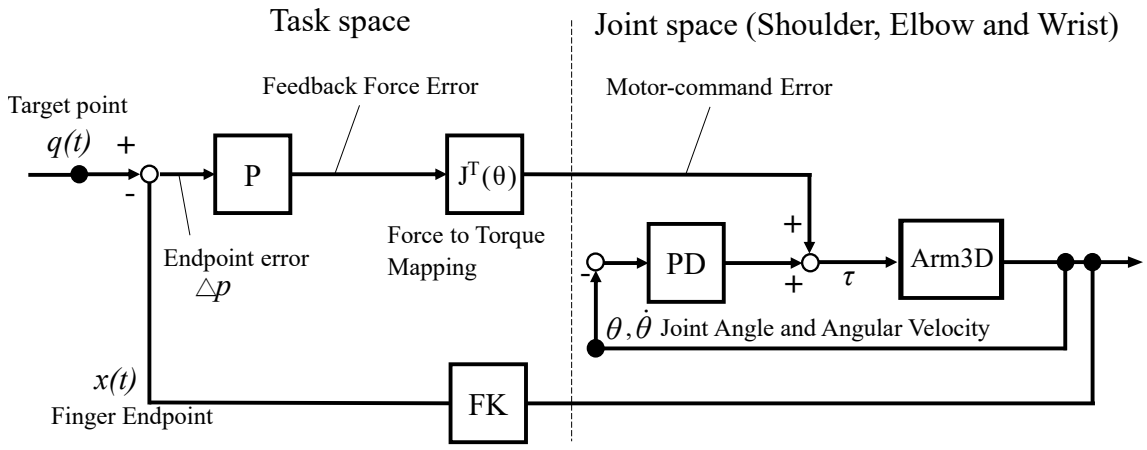


Figure 2.17: The classical PD controller for the Arm3D point tracking tasks. The P block represents the proportional controller while the PD block represents the proportional-derivative controller. The block $J^T(\theta)$ and FK are to map the parameters back and forth between the 3D task space and the Arm3D joint space.

similar to the Arm3D agent. The control loop in Figure 2.17 contains a few different steps. First, the distance between the target point $q(t)$ and the current finger tip position $x(t)$ is calculated, denoted as Δp. The endpoint error then passes through a proportional (P) block to become the feedback force error. The feedback force error is mapped by the inverse Jacobian of the Arm3D agent, the $J^T(\theta)$ block into the motor-command error in the joint space. The output of the local PD controller in the joint space, which contributes to the smooth changes of the joint angles, is added together with the motor-command error and produce input torque commands for the Arm3D's joints. The block FK then maps the resulting joint angle and angular velocity back into the task space as the finger endpoint $x(t)$, and the control loop repeats.

The torque input for the Arm3D agent described above can be written in equation as the Eq. (2.3.5).

$$\tau(t) = -J^T(\theta)k\Delta p - A\theta - B\dot{\theta} \tag{2.3.5}$$

where $\tau(t)$ is the control torque inputs for the joints; θ the joint angels; $\dot{\theta}$ the angular velocity; $J^T(\theta)$ the transpose of the Jacobian of the Arm3D; k the gain of the task space proportional feedback; Δp the endpoint error vector; A and B the diagonal matrices which consist of the proportional and derivative gains of the local PD controller in the joint space.

In contrary to the DRL algorithm which needs to be trained, the PD feedback control does not require any training. For the experiments in this study, the PD

control is tested with 1000 simulation steps, which is corresponding to 20 s, for both the 3-DOF and 7-DOF Arm3D agents.

2.3.1.3 Additional evaluation metrics

To evaluate the results of the Arm3D agents, the previous synergy analysis and the energy equation 2.2.4 are equally used in this study. In addition, a few evaluation metrics are introduced here. First, to evaluate the tracking performance of the Arm3D agent, the average endpoint error during the training steps are calculated, which is the root-mean-square (RMS) error between the target point and the fingertip endpoint throughout one rollout of simulation.

Second, a new measure is introduced to correctly evaluate the rate of motion accuracy per the energetic effort. It is used in [75] to evaluate the coupled index over both error and energy for a reaching task, and it is named as the E-E index, which means (1/Error)/Energy. It is simply the tracking accuracy rate per energy consumption. 1/Error indicates the tracking accuracy, and is a hyperbolic measure used to evaluate a situation with lesser error as a priority. The zero error situation would not occur in the moving-object-following task. This index was used for a posterior evaluation of the performance of the Arm3D agent controlled by both the SAC algorithm and the PD controller.

Lastly, the joint correlation metric characterizes the coordination of the Arm3D's shoulder and elbow joint movements is also introduced. This metric is also employed in neuro-rehabilitation study [76] to measure motor synergies. In this study, for the 3-DOF Arm3D agent, the joint correlations are calculated from the shoulder and elbow joint angles in the sagittal plane. For the 7-DOF Arm3D agent, the joint correlations are computed from the shoulder flexion angle and elbow flexion angle since flexion of both joints are mainly required to perform the circular reaching task in this study.

2.3.2 Experimental Results

2.3.2.1 Performance and Energy efficiency Comparison between DRL and PD controller

Figure 2.18 illustrates the fingertip endpoint transition during the planar tracking task of the 3-DOF Arm3D agent when controlled by: (a) the PD feedback control and (b) the DRL algorithm. On the left of the Figure 2.18, a cool color map,

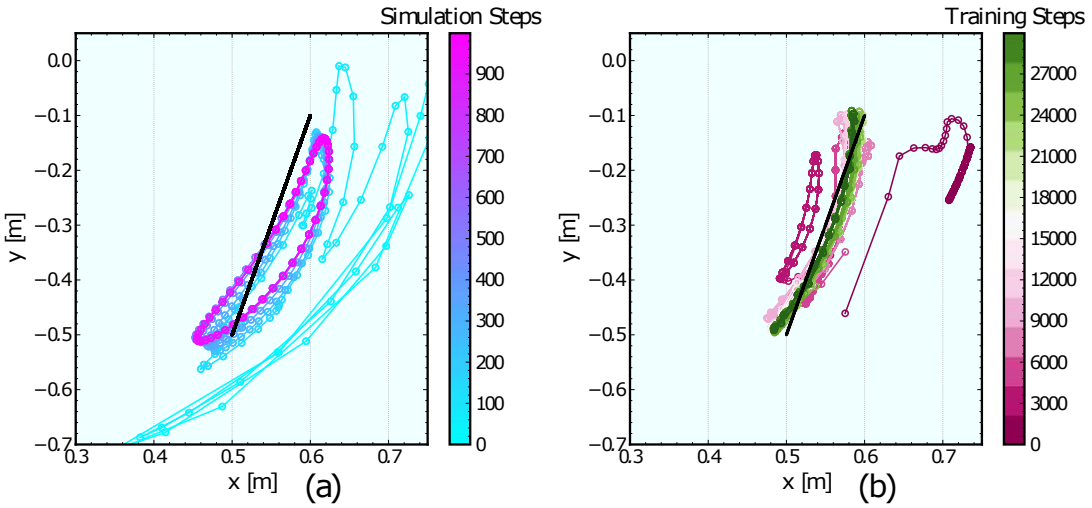


Figure 2.18: The endpoint transitions of 3-DOF Arm3D agent (a) with the PD feedback control and (b) with the DRL algorithm. For the PD controller (left), the color of the curve changes as the simulation step advances. For the DRL algorithm (right), the color of the curves changes as the training step advances. The black solid lines in both plots are the trajectory of the target point.

which illustrates sequential transition of the endpoint, is used for the PD feedback control, while a PiYG color map (ranging from purple to green) is used for the DRL algorithm to illustrate the sequential learning transition. The endpoint transition in the Figure 2.18(a) shows that the PD feedback controller is affected by the gravity (the downward direction of the y-axis) and the interaction torques between joints of the Arm3D agent. Since there is no learning effect for the PD feedback control, the endpoint loop remains unchanged after the initial transitory phase and a constant error exists. By contrast, in the case of the DRL algorithm, it can be noticed that the fingertip trajectory gets closer to the target (black solid line) as the training progresses, reducing the effects of gravity and interaction torque. This qualitative result is already demonstrating the advantage of the DRL algorithm over a classical PD controller, specifically the learning ability of the DRL algorithm.

Next, the quantitative results are presented in the Table 2.1, including the endpoint tracking error, the energy consumption, and the E-E index developments throughout the training phase of the 3-DOF Arm3D agent. As there is no training for the PD controller, only the first row of the table is filled. From this table, it can be noticed that while at the early stage of the training, the performance of the DRL algorithm is worse than the PD controller in terms of the three metrics in the table. However, as the training progresses, the DRL algorithm eventually surpasses the PD controller as it achieves lower tracking error, lower energy consumption and higher energy efficiency in controlling the Arm3D agent.

Table 2.1: Endpoint Tracking Error, Energy Consumption, and E-E Index of the Planar Tracking Task for the 3-DOF Arm3D agent

	PD Feedback Control			DRL Control		
	Error	Energy	*E-E*	Error	Energy	*E-E*
1000	0.041	10.890	2.262	0.224	2.299	1.940
6000				0.045	6.318	3.544
12000				0.022	5.836	7.853
18000	-	-	-	0.019	5.724	9.116
24000				0.017	5.684	10.290
30000				0.016	5.581	10.749

Figure 2.19 illustrates the top view of fingertip endpoint transition during the circle drawing tasks for the 7-DOF Arm3D agent when controlled by: (a) the PD feedback control and (b) the DRL algorithm. Similarly to the previous task, there is a constant tracking error for the PD controller. For the DRL algorithm on the right of the Figure 2.19, the final tracking error after the training is clearly lower than that of the PD controller. However, the initial fingertip endpoint transition of the DRL algorithm is more random than the PD controller due to the exploration for a better solution at the early learning phase. This is one of the key characteristic of a DRL algorithm.

The quantitative results for the 7-DOF Arm3D agent is presented in the Table 2.2, including the endpoint error, the energy consumption, and the E-E index developments throughout the training. From this table, it can equally be noticed

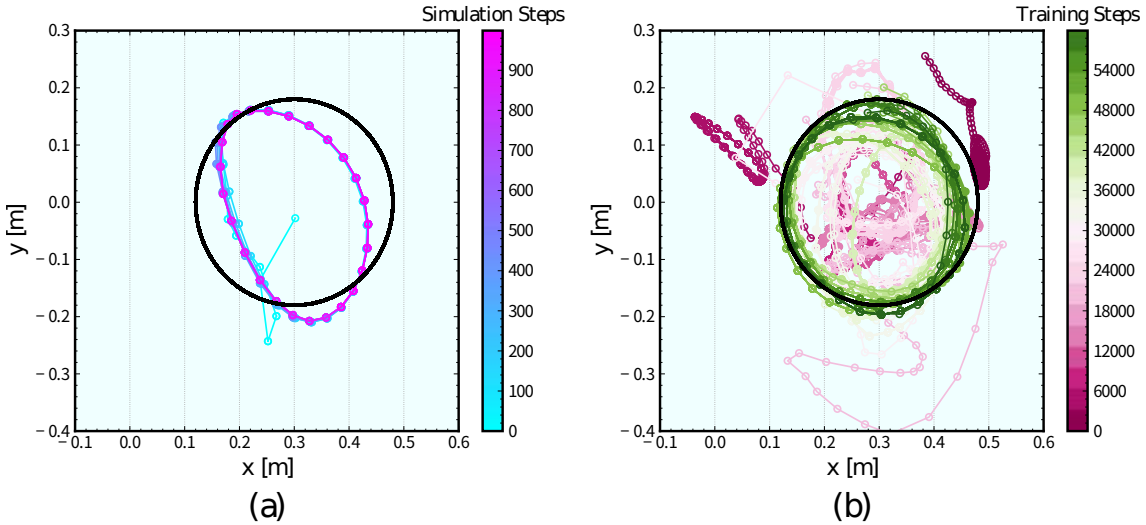


Figure 2.19: The endpoint transitions of 7-DOF Arm3D agent (a) with the PD feedback control and (b) with the DRL algorithm. For the PD controller (left), the color of the curve changes as the simulation step advances. For the DRL algorithm (right), the color of the curves changes as the training step advances. The black solid circles in both plots are the trajectory of the target point.

Table 2.2: Endpoint Tracking Error, Energy Consumption, and E-E Index of the Planar Tracking Task for the 7-DOF Arm3D agent

	PD Feedback Control			DRL Control		
	Error	Energy	*E-E*	Error	Energy	*E-E*
1000	0.086	8.771	1.326	0.249	1.222	3.286
10000				0.137	14.612	0.501
20000				0.164	20.542	0.297
30000	-	-	-	0.089	19.528	0.574
50000				0.035	16.978	1.666
60000				0.026	17.903	2.140

that after training, the performance of the DRL algorithm is better than the PD controller in exception of the energy consumption metric. This can be due to the existent of extra joint rotations for minor improvement of the tracking error. Despite the higher energy consumption, the energy efficiency E-E of the DRL algorithm is still higher than the PD controller.

2.3.2.2 Synergy Emergence Phenomenon

Figures 2.20 and 2.21 illustrate the phase portraits between the shoulder and elbow joint angles of the 3-DOF and 7-DOF arm3D agents respectively, where on the left of the figures are for the PD controller and the right of the figures are for the DRL algorithm. The phase portraits are time-independent and they shows how well the two joints coupled to each other in the working space. The joint correlation coefficient is also calculated and shown in the top left of each plot to indicate how closely coupled are the joints. For both the 3- and 7-DOF agents, the PD controller shows more unrelated and non-synergetic solutions between the shoulder and elbow as the joint correlation coefficient is lower than the DRL algorithm. Indeed, the DRL solutions consistently converge to an aligned joint combinations during learning,

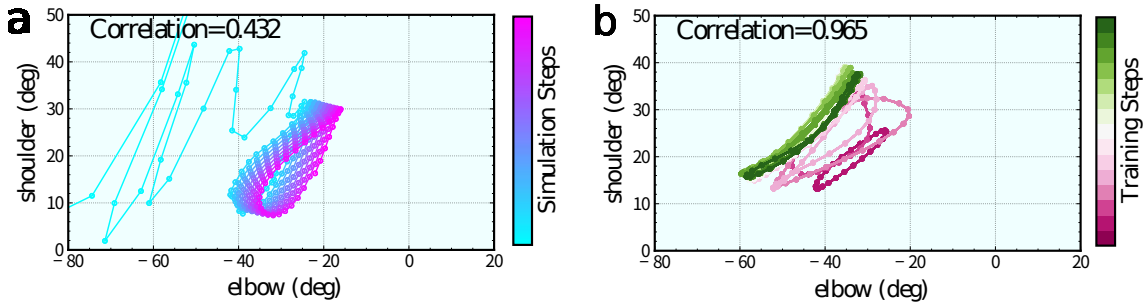


Figure 2.20: The phase portrait of the 3-DOF Arm3D agent between the shoulder and the elbow joint angles for (a) the PD controller and (b)the DRL algorithm. The coefficient of the joint correlation in the final phase is shown in the upper left corner of each plot as a metric of the joint synergy.

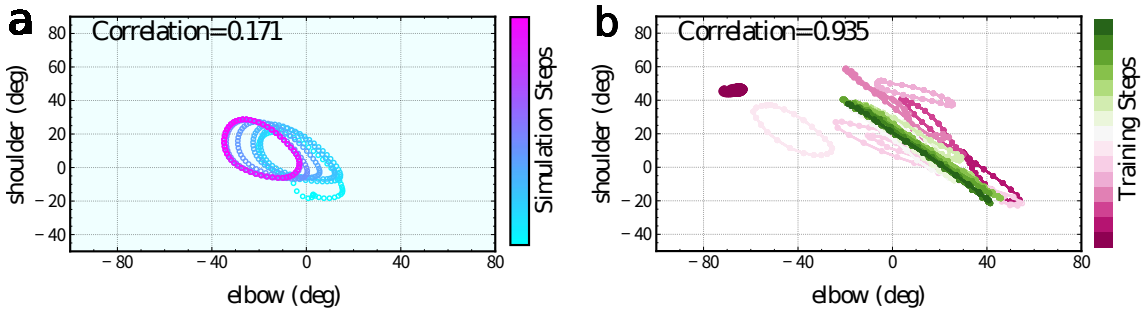


Figure 2.21: The phase portrait of the 7-DOF Arm3D agent between the shoulder and the elbow joint angles for (a) the PD controller and (b)the DRL algorithm. The coefficient of the joint correlation in the final phase is shown in the upper left corner of each plot as a metric of the joint synergy.

indicating that the shoulder and elbow motions become tightly coupled to execute the given tasks. This indicates that the joint synergy emerges through the learning phase for the DRL algorithm.

As a final confirmation to show the synergy emergence phenomenon in the DRL learning process, the synergy development graph introduced in the first part of this chapter is plotted. Figure 2.22 shows the synergy development graphs for both the 3-DOF Arm3D agent (left) and the 7-DOF Arm3D agent (right). As one can notice on the graphs, the synergy level increases as the learning progresses, indicating more coordinated movements are carried out to accomplish the tracking tasks. These results match well with the joint correlation coefficient results.

2.4 Discussion

As mentioned before, the synergy concept is not explicitly implemented in SAC, however, it can be observed in both studies of the chapter that it converges to the optimal cases by arriving at a synergetic solution which is similar to the human learning case. This ultimately suggests that the synergy concept is indeed needed in a learning process for an agent with redundant joints to achieve a task in an optimal way [69][75]. Indeed, SAC and TD3 are applicable for a lot of different problems other than for robotics, which suggests that a highly-specialized learning algorithm with the synergy concept integrated into it may be more suitable for robotic tasks and further improve the performance and energy efficiency achieved by SAC and TD3. The importance of energy efficiency is mentioned in [41][42] as the energy consumption issue is no longer negligible in a real-world robot.

For the current work, it is the first step to try to apply the synergy concept in analyzing DRL algorithms and the joint synergy analysis is only done on simulated

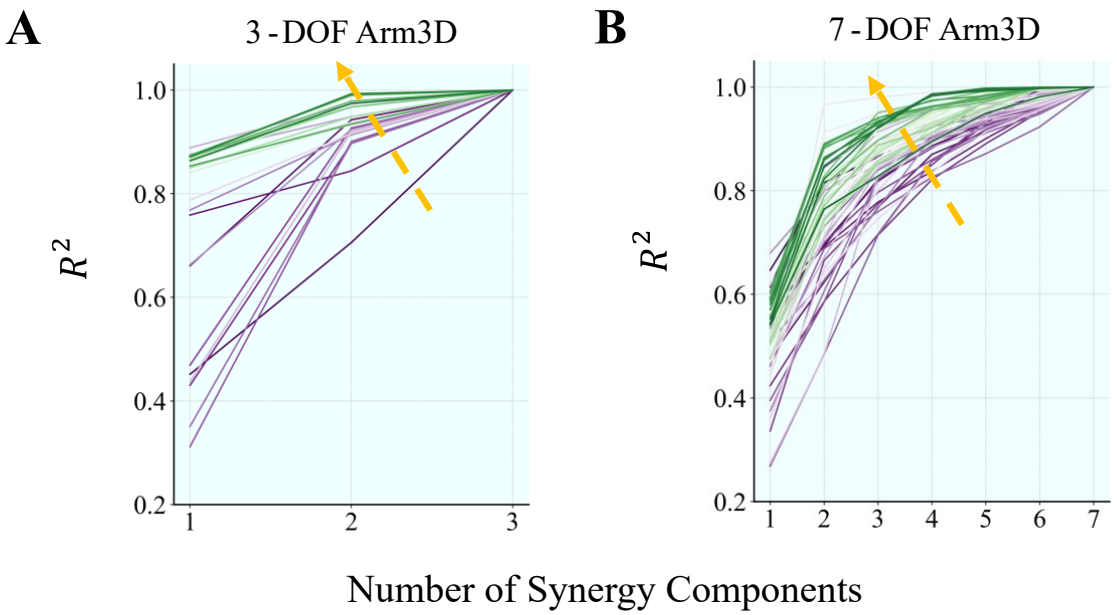


Figure 2.22: The synergy development graph for the 3-DOF Arm3D agent on the left and the 7-DOF Arm3D agent on the right. The purple curves correspond to the early training phase while the green curves correspond to the end of the training phase. The fewer number of curves for the 3-DOF Arm3D graph is due to the fewer training steps needed for the easier 2D planar tracking task. Both the graphs show that the synergy level increases throughout the DRL training phase.

agents. There is still room for improvement in both the studies. For example, in the synergy extraction process, low energy simple movements can a have high synergy level (but low performance), and this may make the results harder to interpret. Another issue is that while the physics engine MujoCo [67] and OpenAI gym [68] may simulate complex dynamics for the locomotion task, there could be some gap from the real-world dynamics. As a result, more future works need to be done on more complex agents and environments. This chapter serves as a preliminary step to show that the synergies are indeed related to the improvement in performance and energy efficiency during the learning phase of DRL algorithms.

2.5 Conclusion

The analysis in this chapter has tried to link two different fields of research, which is the human motor synergy concept and the DRL for robotics, and demonstrated some promising relationships between the two domains. Although the synergy constraint has never been encoded into the reward function of a DRL algorithm, the synergy emergence phenomenon could be observed statistically in the learning agent in both studies in this chapter. To our knowledge, it is the first attempt to quantify the synergy development in detail and evaluate its emergence process during

deep learning motor control tasks. Indeed, it has been successfully shown that the synergy level correlates overall well with the performance and the energy efficiency metrics for the SAC-trained agents, while for the TD3-trained agents, the synergy level remains constant or decreases even if the two metrics are gradually increasing. Interestingly, the proposed synergy-related metrics equally reflected a better learning capability of SAC over TD3. It suggests that these metrics could be additional new indices to evaluate DRL algorithms for motor learning. As a result of this study, it has been demonstrated that the synergy concept may be one of the key elements for DRL algorithms and one should consider to include synergy constraints when designing a new DRL algorithm for robotics. This may speed up the learning process of a robotic agent with better performance and energy efficiency. The results also indicated that the synergy of the body control is an issue related to the coupling of performance and energy, as proposed in some previous studies of the coordination of a redundant robotic arm [75][77]. In the second part of this chapter, it has also been shown that the learning ability of a DRL algorithm is one of the advantages that it is better than a classical PD controller. This ability enables DRL algorithm to achieve higher performance and more energy-efficient solutions than the PD controller, besides confirming the synergy emergence phenomenon in DRL, as shown by the results of the Arm3D point tracking tasks.

Chapter 3

Deep Reinforcement Learning for Joint Redundancy Quantification

3.1 Introduction

In the previous chapter, a synergy analysis is carried out on simulated redundant agent using DRL and several synergy-related metrics are introduced. One metric in particular, the absolute surface area (ASA) of the synergy development, is found to be capable of quantifying the amount of exploration performed by a DRL algorithm before arriving at an optimal control policy to accomplish a given task. When the algorithm needs to explore excessive suboptimal sequences of joint movements during the training phase before finding an optimal solution, this typically indicates that the robotic agent has a high degree of joint redundancy, and vice versa. Indeed, the excessive exploration required before accomplishing the primary task is a manifestation of high joint redundancy, as by definition, a redundant robot is one that possesses more resources than those strictly required to execute its primary task [56].

Applying the above concept in the study in this chapter, the ASA metric is used to quantify the relative joint redundancy of a robotic agent between different joint configurations. In the following of this study, the ASA metric is renamed to the synergy exploration area (SEA) metric, which is more appropriate for the context in this study. Indeed, synergy [55] can be interpreted as the mode of motion of a robotic agent. As a DRL algorithm explores different modes of motion to find an optimal policy for the given task, it indirectly conveys information about the robotic joint redundancy.

Motivated by the possibilities that DL offers in terms of scaling to difficult problems, DRL is used to approach the problem of quantifying robotic joint redundancy. To further show that the degree of redundancy is more than just counting the number of joints a robotic agent has, further experiments are carried out to use the SEA metric for quantifying the joint redundancy affected by the kinematic properties and the dynamic properties of an agent. Fig. 3.1 gives a quick glance at the simulated robotic agents and the corresponding tasks used in our study. Detailed explanations are given further in this chapter.

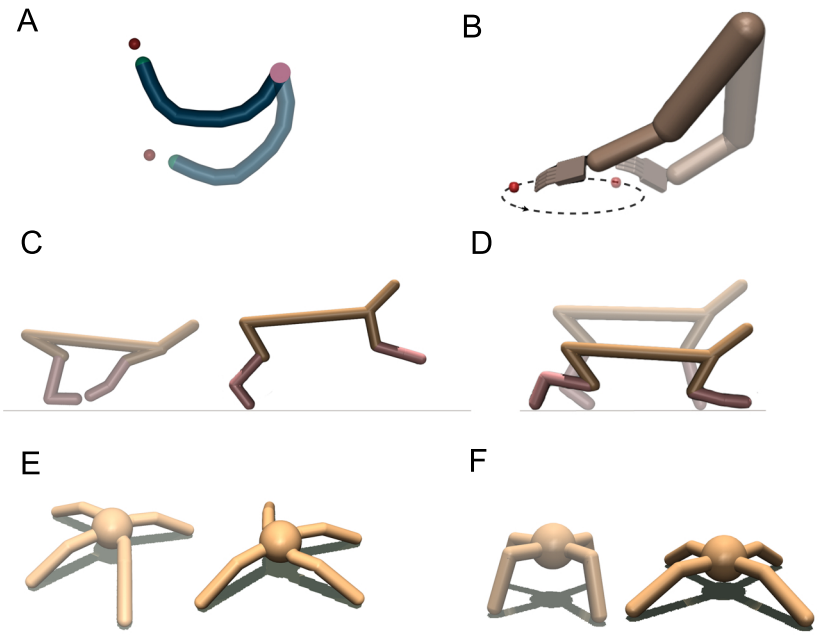


Figure 3.1: The simulated robotic agents with their respective task(s) used in our joint redundancy quantification study. (A) Planar line trajectory-following task for the Arm2D agent with a red mobile target; (B) 3D circular trajectory-following task for the Arm3D agent; (C) running task for the Half-Cheetah; (D) squatting task for the Half-Cheetah; (E) running task for the Ant; (F) squatting task for the Ant.

3.2 Method

3.2.1 Configurations of simulated agents

A total of four different types of simulated agents are investigated in this chapter as shown in Fig. 3.2. These simulated agents were simulated using a simulation engine named MuJoCo [78] with specific physical properties such as the body segment weights and dimensions. The physical properties of the robotic agents are made sure to be reasonable and a feasible solution could be found by the DRL algorithms considering the dynamic condition of the agents.

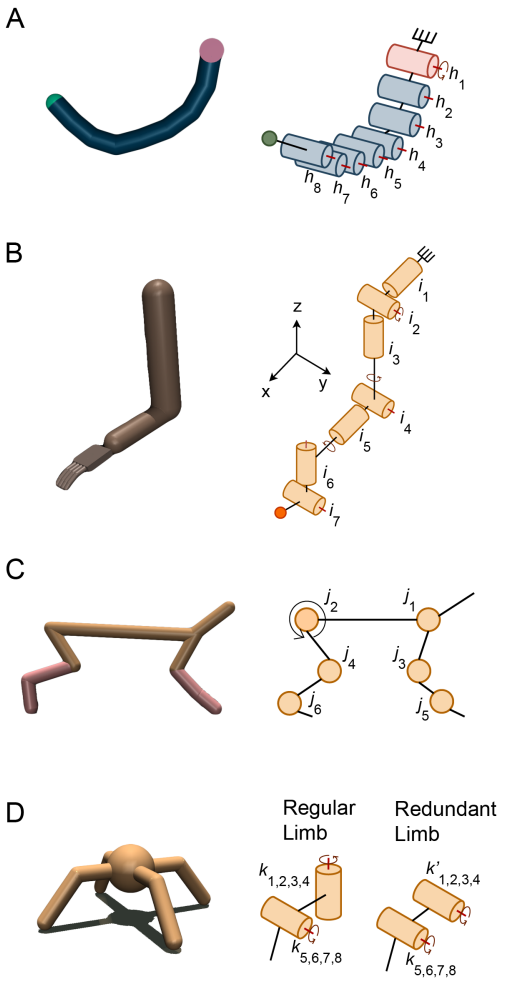


Figure 3.2: Simulated robotic agents with their corresponding kinematic diagrams. All joints are hinge joints and each joint is annotated with a letter and a subscript number: (A) Arm2D, a simplistic multi-joint robotic arm for following planar trajectories. The green extremity is the fingertip. (B) Arm3D, a human-like robotic arm with seven degrees of freedom for following circular trajectories in the three-dimensional (3D) space. Joints i_1 to i_3 enable shoulder abduction, flexion, and rotation, respectively; i_4 and i_5 enable elbow flexion and pronation, respectively; and i_6 and i_7 enable wrist flexion and abduction, respectively. The orange extremity is the fingertip. (C) Half-Cheetah, a robotic agent that mimics the limb configuration of a half-side of a quadruped; (D) Ant, a robotic agent with four limbs having two joints each. There are two types of limb, namely the regular limb and the redundant limb, that are used in two different experiments. The hip joints are annotated k_1 (k'_1) to k_4 (k'_4) and the knee joints are annotated k_5 to k_8.

First, a simplistic multi-joint robotic arm, the Arm2D agent (Fig. 3.2A) is used as a toy example to demonstrate that the SEA metric can effectively measure the relative joint redundancy in the simplest case. The Arm2D is designed to follow a planar line trajectory (Fig. 3.1A), and the number of joints aligned in parallel is varied from two to eight in these experiments by removing certain joints (Table 3.1), with the remaining joints always distributed equidistantly between the pink shoulder joint and the green fingertip. In theory, two degrees of freedom (DOF) are enough for tracking the planar trajectory in the 2D space. This toy example allows us to isolate the joint redundancy aspect of interest without the need to consider other non-essential aspects. Additionally, a more realistic human-like robotic arm, the Arm3D agent (Fig. Fig. 3.2B) is used to illustrate that the proposed method can be scaled to a more complicated case. This agent is assigned to perform the circle drawing task in the 3D space (Fig. 3.1B). This task demands at least three DOF for the position tracking, four DOF when the hand orientation is maintained, and the Arm3D is well-suited for this task. The Arm3D has seven DOF, with three DOF assigned to the shoulder for abduction, flexion, and rotation; two DOF assigned to the elbow for flexion and pronation; and two DOF assigned to the wrist for abduction and flexion. To investigate the relative joint redundancy of the Arm3D under different configurations, the less important joints were gradually added to a 3-DOF Arm3D agent (Table 3.1) in the experiments, thus increasing the degree of joint redundancy. The aim of these experiments is to verify that the SEA metric can provide relevant information for the added joint redundancy in the Arm3D agent as it executes a circular tracking task. Indeed, the joint configurations in Table 3.1 are designed such that the most crucial joints for the given task will always be present, with the less important joints gradually added to highlight the increase in joint redundancy, if any occurs.

To further extend the study to uncommon cases, two additional robotic agents are investigated; the Half-Cheetah (Fig. 3.2C) and the Ant (Fig. 3.2D). These two agents have unconventional robotic structures, and classical model-based approaches to quantify their joint redundancy will easily fail as the dynamically feasible redundant solution space is not easily computed for the free-moving robots. Indeed, different from the Arm2D and Arm3D agents whose shoulder joints are anchored at a fixed point, the Half-Cheetah and Ant agents can move freely in the 2D and 3D space respectively interacting with the environment. When a robot base is fixed at a reference point, the dynamic feasibility depends solely on the body side dynamics condition, free from the environmental interaction. However, for the free-moving robots, the state of the contact is not static, then the redundancy of dynamically feasible solution is difficult to be quantified in a systematic way.

Table 3.1: Joint configurations for varying the DOF of each agent

DOF	Arm2D	Arm3D	Half-Cheetah	(Redundant-) Ant
2	h_1, h_5	-	$j_1 - j_2$	-
3	-	$i_2 - i_4$	-	-
4	$h_1, h_3,$ h_5, h_7	$i_1 - i_4$	$j_1 - j_4$	-
5	-	$i_1 - i_4, i_6$	-	-
6	$h_1, h_3,$ $h_5 - h_8$	$i_1 - i_4,$ $i_6 - i_7$	$j_1 - j_6$	-
7	-	$i_1 - i_7$	-	-
8	$h_1 - h_8$	-	-	$(k'_1 - k'_4)/k_1 - k_4,$ $k_5 - k_8$

The Half-Cheetah and the Ant structures serve as the first verifications that our DRL-based joint redundancy quantification approach can scale to complex robotic agents. First, a running task (Fig. 3.1C) is considered for the Half-Cheetah with two-, four- and six-joint configurations (Table 3.1). Then, the task is changed to a squatting task (Fig. 3.1D), i.e., the Half-Cheetah must try to reach the lowest and highest reachable points alternatively with respect to the center of its body. By performing these two sets of experiments with different tasks, we aim to demonstrate that the SEA metric can not only provide information about the relative joint redundancy of an agent for a single task, but also about functions across different tasks. From this perspective, a running (Fig. 3.1E) and a squatting task (Fig. 3.1F) are similarly designed for the Ant with different joint configurations (Table 3.1). Specifically, the four hip joints of the Ant are first set to enable abduction, i.e., rotation around the vertical axis at the center of its body, within a fixed range. This configuration of limbs is denoted as the regular limbs (Fig. 3.2D). Then, exclusively for the squatting task, the hip joints are configured to enable flexion within a small range to demonstrate that the direction of rotation of a joint can affect the overall joint redundancy for a task. This type of limbs is named as the redundant limbs (Fig. 3.2D) and the corresponding agent as the Redundant-Ant. Indeed, the joint redundancy does not depend solely on the number of joints, but also on the kinematic properties of the joints for the given task. The SEA metric is used in

hope of capturing this subtle aspect through these experiments. To further show the usefulness of the SEA metric, it is also tested to verify if it can quantify the joint redundancy affected by the kinemtaic properties and the dynamic properties of an agent, which is explained later.

3.2.2 Task specification in Deep Reinforcement Learning

The DRL algorithms employed in all the experiments are the SAC [79] algorithm and the TD3 [80] algorithm. The reward functions for the DRL algorithms allow us to specify the tasks for each agent. For the Arm2D and Arm3D agents, the reward function is the distance between the fingertip and the red target point (Fig. 3.1A, 3.1B). The reward function can be written as the Eq. (3.2.1), where the positions of the finger tip and the target point are represented by the vectors $\overrightarrow{fingertip}$ and $\overrightarrow{target}$ respectively. These vectors are in function of the Cartesian coordinates (x, y, z) and the time t. Besides the distance penalty in the reward function, a small penalty on the total torque usage of all joints are included, written as the term $\sum_i A_i(t)^2$ in the Eq. (3.2.1). This term is to encourage energy efficient movement of the trained agent, and it is commonly included by default in most DRL trainings. The two terms in the Eq. (3.2.1) are scaled to give more weight to the target following objective, with a small consideration for energy saving.

$$\begin{aligned} R(t) = &- 5 \cdot ||\overrightarrow{fingertip}(x, y, z, t) - \overrightarrow{target}(x, y, z, t)||_2 \\ &- 0.05 \cdot \sum_i A_i(t)^2 \end{aligned} \tag{3.2.1}$$

For the running task with the Half-Cheetah and the Ant, the reward function includes mainly the forward speed of the agent at a given time. The Eq. (3.2.2) is the detailed reward function for the Half-Cheetah and the Eq. (3.2.3) is for the Ant. Besides the forward speed reward and the energy saving penalty, there is an additional reward for staying upright for the Ant, as shown in the Eq. (3.2.3) as the *healthy_reward*. This reward is added for the Ant as it is found that the Ant will easily overturn if this reward is not added.

$$R(t) = \overrightarrow{v(t)} - 0.1 \cdot \sum_i A_i(t)^2 \tag{3.2.2}$$

$$R(t) = \overrightarrow{v(t)} - 0.5 \cdot \sum_i A_i(t)^2 + healthy\_reward \tag{3.2.3}$$

For the squatting tasks for both the Half-Cheetah and the Ant, a red target point is defined such that it alternates between the highest and the lowest points aligned vertically with the agent's center of gravity. Therefore, the reward function for the squatting task is the distance between the agent's center of gravity and the target point, as shown in the first term of the Eq. (3.2.4). An additional reward, i.e. the *horizontal_reward*, is added to encourage the agent to stay horizontally flat while squatting. Similarly to the running task of the Ant, a *healthy_reward* is also added to prevent the agent from overturning at the early phase of training.

$$\begin{aligned} R(t) = & -5 \cdot ||\overrightarrow{body}(x,y,z,t) - \overrightarrow{target}(x,y,z,t)||_2 \\ & + 0.1 \cdot horizontal\_reward \\ & + healthy\_reward \end{aligned} \tag{3.2.4}$$

3.3 Experimental Results

3.3.1 SEA for joint redundancy quantifications

During the experiments, there were several parameters that must be determined to calculate the SEA metric. The first parameter is the total number of training checkpoints for each agent. 20 to 30 training checkpoints are used in this study depending on the complexity of the robotic agents, with each checkpoint saved periodically throughout the DRL training process. The details of the choice of the total number of training checkpoints for each agent are summarised in the Table 3.2. The second parameter is the time window size for the joint torque signal sampling. It is necessary to choose a window size that sufficiently covers the essential part of the signals during a task. For the running task, the window size is determined by measuring the time needed for a robotic agent to run a fixed distance. For other tasks, a window size was chosen such that it was sufficiently long to cover a few cycles of the agent's joint motions during a given task.

Table 3.2: Training parameters for each agent in the experiments

Robotic agents	Tasks	DOF	Training iterations	Iteration(s) per checkpoint
Arm2D	Path-following	2	30	1
		4	30	1
		6	390	13
		8	390	13
Arm3D	Path-following	3	90	3
		4	90	3
		5	90	3
		6	90	3
		7	90	3
Half-Cheetah	Running	2	3000	100
		4	3000	100
		6	3000	100
	Squatting	2	120	4
		4	120	4
		6	120	4
Ant	Running	8	300	15
	Squatting	8	500	20
Redundant-Ant	Squatting	8	500	20

Before analyzing the capability of the proposed DRL-based joint redundancy quantification approach, the task performance and the SEA development during the training progress of all the agents with different configurations and different tasks are plotted. The Fig. 3.3 and the Fig. 3.4 show the training progress for the SAC algorithm, while the Fig. 3.5 and the Fig. 3.6 show the results for the TD3 algorithm. The solid curves are the average values of five trials of each experiment, and the standard deviation of these five trials are plotted as the shaded region around the average curves. As one can notice in the these figures, as the DRL algorithms explore for better solutions in the early phase of the training, the task performances increase rapidly together with the SEA metric. When the SEA metric does not show much changes or has converged, the task performance will equally be constant or slowly increasing. This validates that the SEA metric can quantify the exploration of the DRL algorithms as demonstrated in the previous chapter. However, one interesting new remark is that when the DOF of an agent is lower, the SEA metric is also lower and the task performance also converges much faster than the agent with higher DOF. This result can be noticed in all of the plots of all agents and different DRL algorithms (SAC or TD3), indicating that this is not a coincidence and it is consistent across all cases. This serves as the first validation step of the proposition that the DRL-based joint quantification using the SEA metric has high possibility to work as expected.

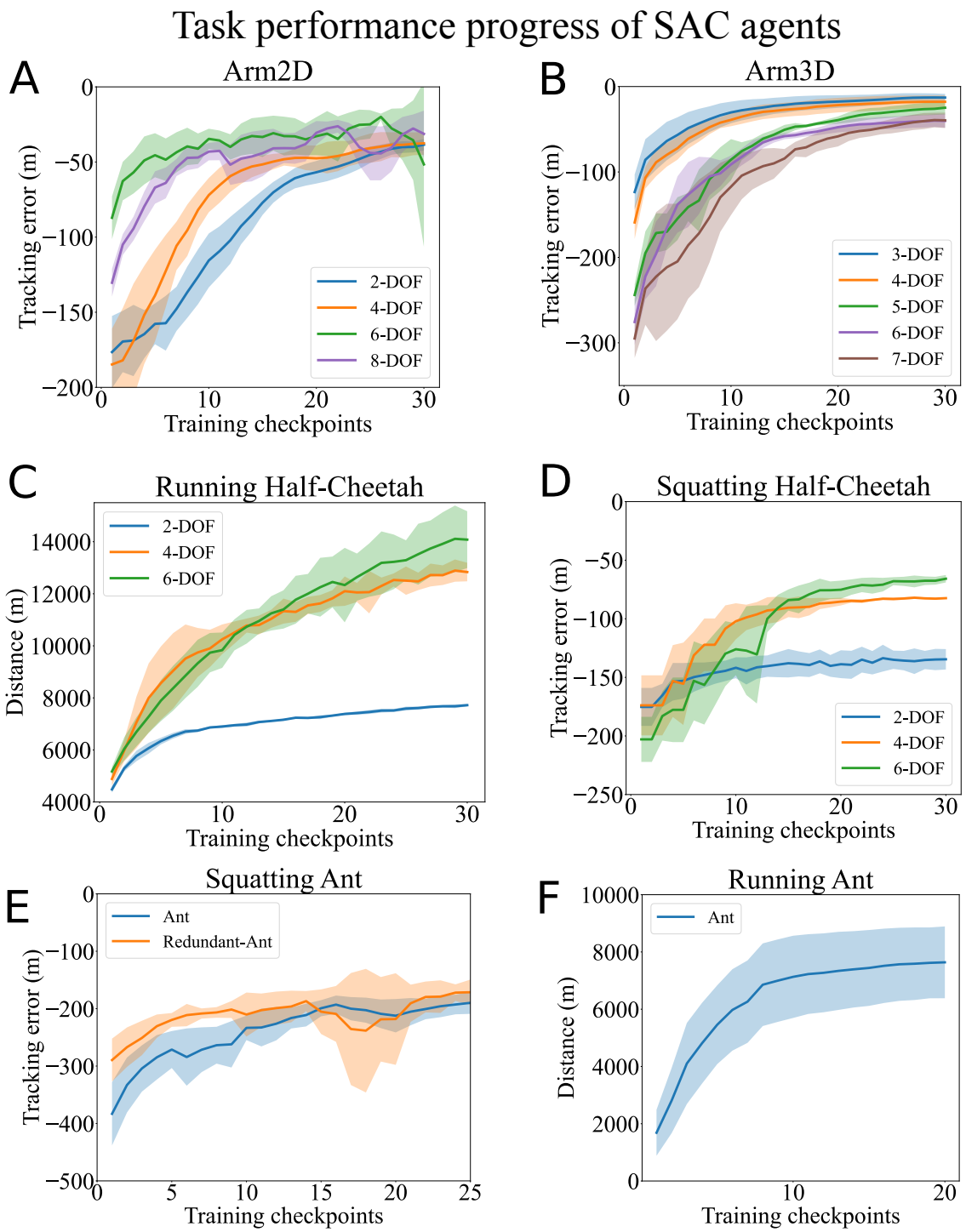


Figure 3.3: Training progress of SAC algorithm under different agent configurations. Standard deviation of the task performance is represented as shaded area around each curve. (A) Training progress of the Arm2D agent with the planar trajectory following task under different joint configurations. (B) Training progress of the Arm3D agent with the 3D circle drawing task under different joint configurations. (C) Training progress of the Half-Cheetah agent with the running task under different joint configurations. (D) Training progress of the Half-Cheetah agent with the squatting task under different joint configurations. (E) Training progress of the Ant and Redundant-Ant agents with the squatting task. (F) Training progress of the Ant agent with the running task.

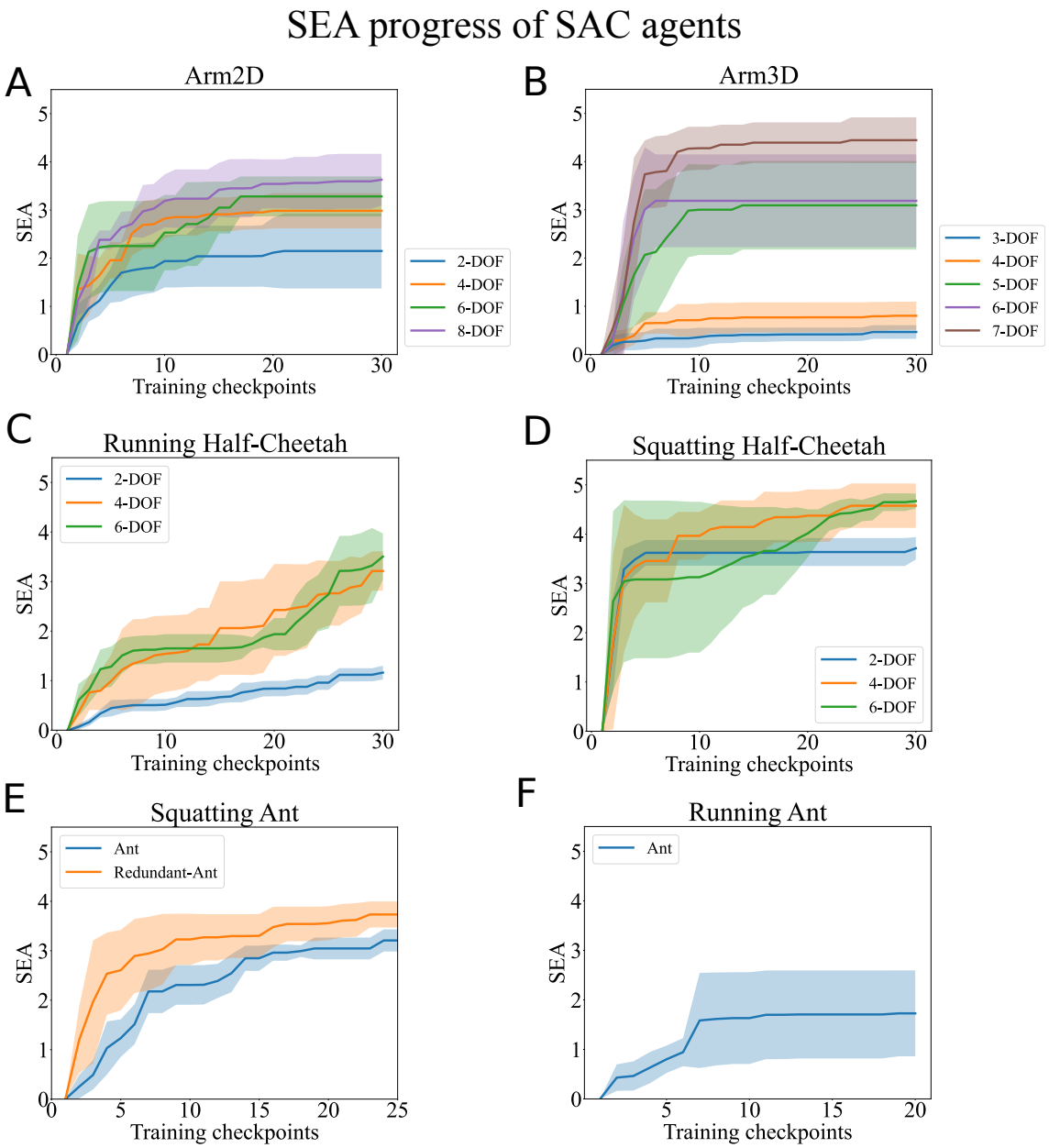


Figure 3.4: Training progress of TD3 algorithm under different agent configurations. Standard deviation of the task performance is represented as shaded area around each curve. (A) Training progress of the Arm2D agent with the planar trajectory following task under different joint configurations. (B) Training progress of the Arm3D agent with the 3D circle drawing task under different joint configurations. (C) Training progress of the Half-Cheetah agent with the running task under different joint configurations. (D) Training progress of the Half-Cheetah agent with the squatting task under different joint configurations. (E) Training progress of the Ant and Redundant-Ant agents with the squatting task. (F) Training progress of the Ant agent with the running task.

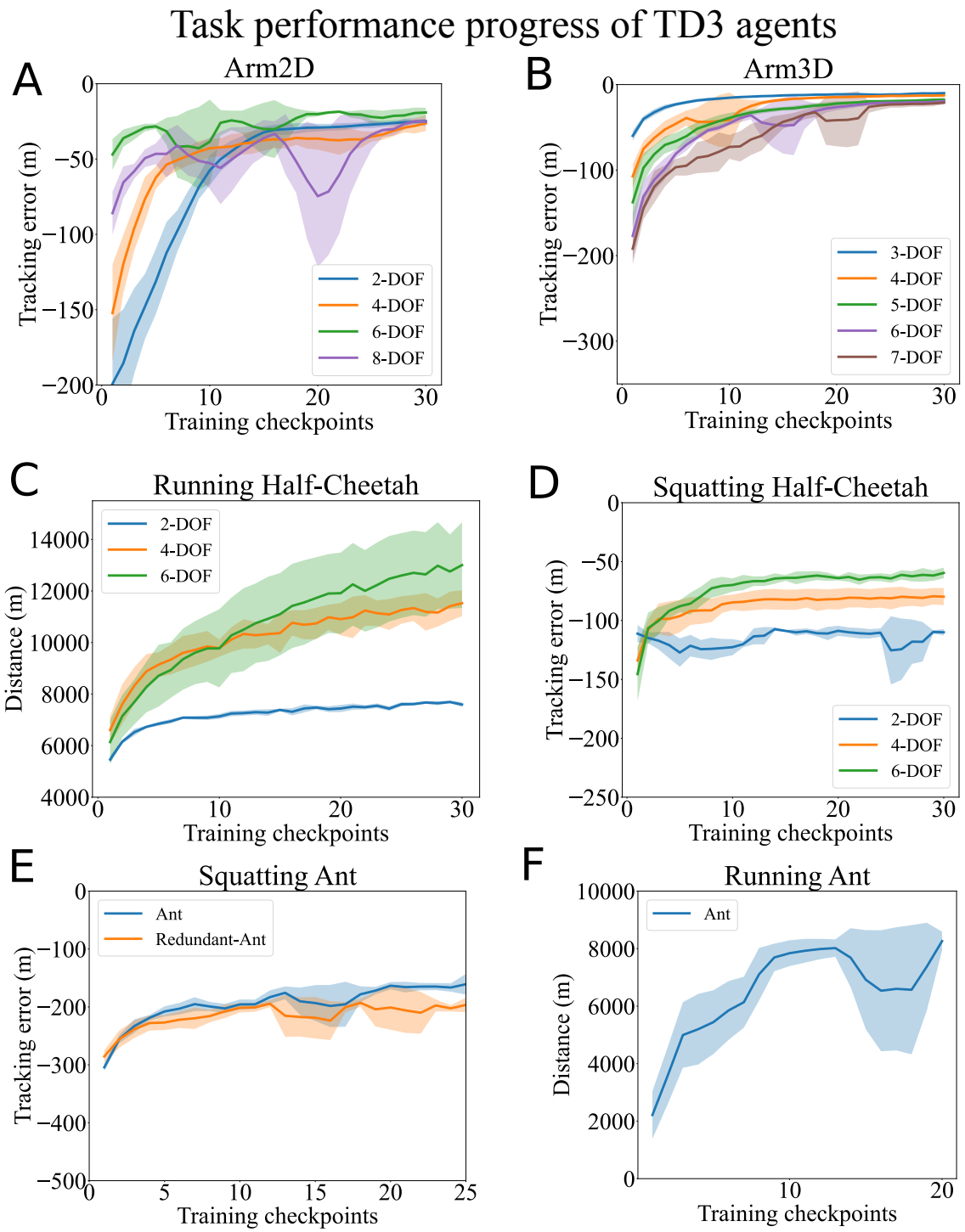


Figure 3.5: Training progress of TD3 algorithm under different agent configurations. Standard deviation of the task performance is represented as shaded area around each curve. (A) Training progress of the Arm2D agent with the planar trajectory following task under different joint configurations. (B) Training progress of the Arm3D agent with the 3D circle drawing task under different joint configurations. (C) Training progress of the Half-Cheetah agent with the running task under different joint configurations. (D) Training progress of the Half-Cheetah agent with the squatting task under different joint configurations. (E) Training progress of the Ant and Redundant-Ant agents with the squatting task. (F) Training progress of the Ant agent with the running task.

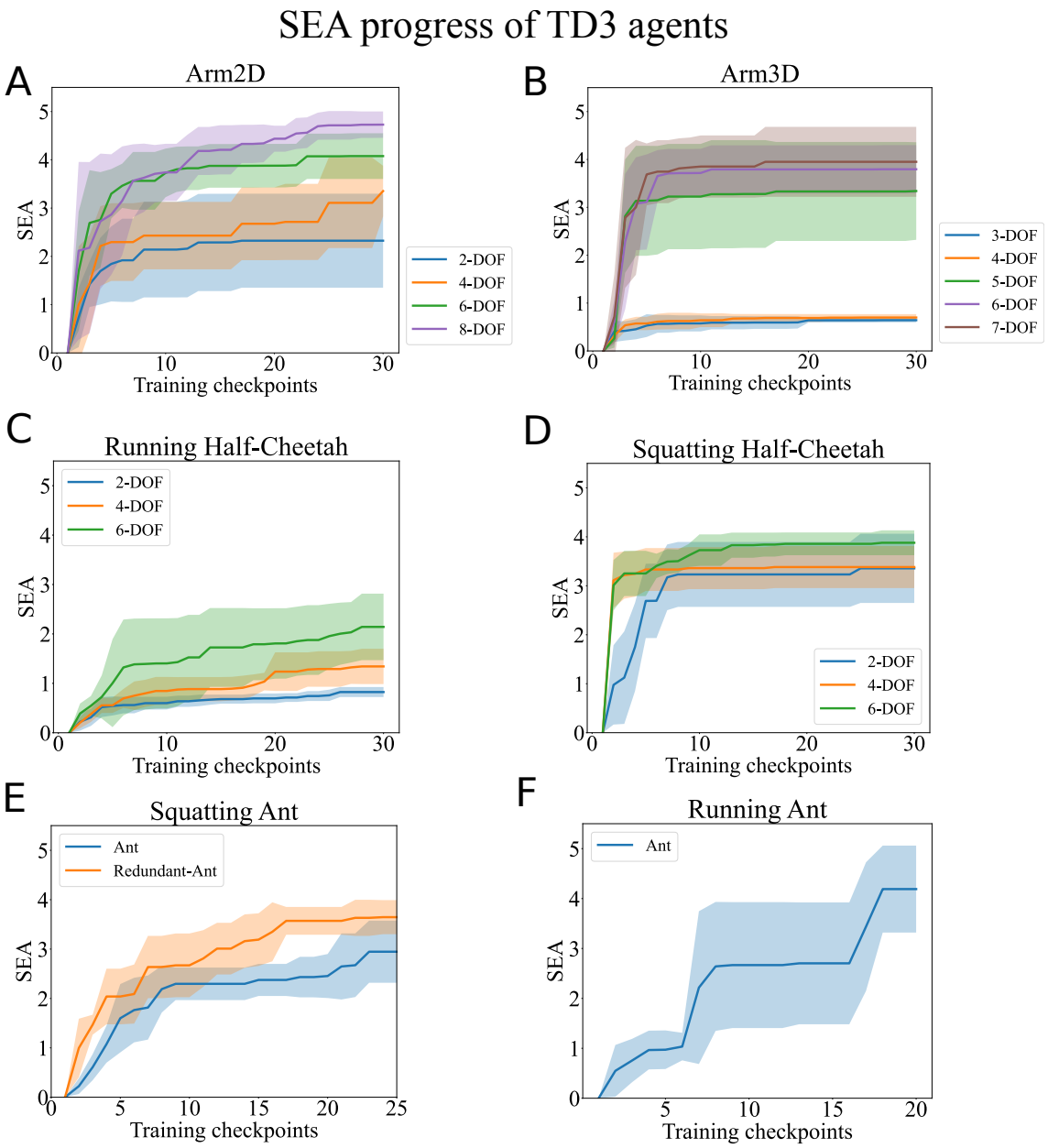


Figure 3.6: SEA metric evolution of TD3 algorithm under different agent configurations. Standard deviation of the SEA is represented as shaded area around each curve. (A) SEA of the Arm2D agent during the training phase under different joint configurations. (B) SEA of the Arm3D agent during the training phase under different joint configurations. (C) SEA of the running Half-Cheetah agent during the training phase under different joint configurations. (D) SEA of the squatting Half-Cheetah agent during the training phase under different joint configurations. (E) SEA of the squatting Ant and Redundant-Ant agents during the training phase. (F) SEA of the running Ant agent during the training phase.

Having observed the task performance and the SEA metric development during the training phase of all agents, it is necessary to investigate in details the capability of the proposed DRL-based joint redundancy quantification approach. To this end, the Fig. 3.7 is plotted to compare the end-of-training SEA values between agents of different configurations. All of the bar plots are constructed by averaging the results of five experimental trials for each agent configuration. The standard deviation of each bar is presented on top of the bar. The green bars are the results for the SAC algorithm and the brown bars are for the TD3 algorithm. Beginning with the Arm2D experimental results (Fig. 3.7A), it can be observed that as the number of joints increases, the SEA also increases in a quasi-linear fashion. As all the added joints are of the same type, one can expect that the amount of redundancy space added by introducing each extra joint should be similar. Thus, the quasi-linear increase of SEA matches well with the logical expectation of how the redundancy space should be increased. This result is obtained purely by the computational exploration of the joint space, with the joint type added being totally unknown to the DRL algorithms. This justifies that the SEA metric can quantify the added redundancy accurately. The quasi-linearity of the increase in the SEA also indicates that all of the joints play an equal role in the trajectory-following task. As a result, the absence of one joint can be replaced by another joint in the two-dimensional (2D) task-space, as long as the Arm2D agent possesses at least two joints, which is the minimum number required to accomplish the 2D task.

The increase in SEA as the joint redundancy increases can also be observed in the Arm3D experimental results (Fig. 3.7B), but with some subtleties. The SEA results suggest that there is a sudden increase in the degree of joint redundancy for the 3D circle-drawing task when the DOF increases from four to five. This transition in DOF corresponds to the addition of the i_6 joint, which is the wrist joint that enables wrist flexion. Indeed, the bar plot is very informative as it reveals that the i_6 joint is not essential for the trajectory-following task. The addition of this joint causes excessive exploration in the DRL optimization process, which is represented by the higher SEA value. In contrast, the DRL algorithm can find a solution for the three- and four-DOF cases without extra exploration, as indicated by the low SEA values. This indicates that joints i_1 to i_4 are the main joints for accomplishing the primary task, and any extra joints, i.e., joints i_5 to i_7 are non-essential for the primary task and can be potentially eliminated if the extra joint redundancy is not desired. Compared with the Arm2D agent, the joints of the Arm3D agent do not exhibit the same importance in the trajectory-following task, as indicated by the SEA results. This outcome is critically important, as we know that the elbow pronation or the wrist movements can be excluded in a trajectory-

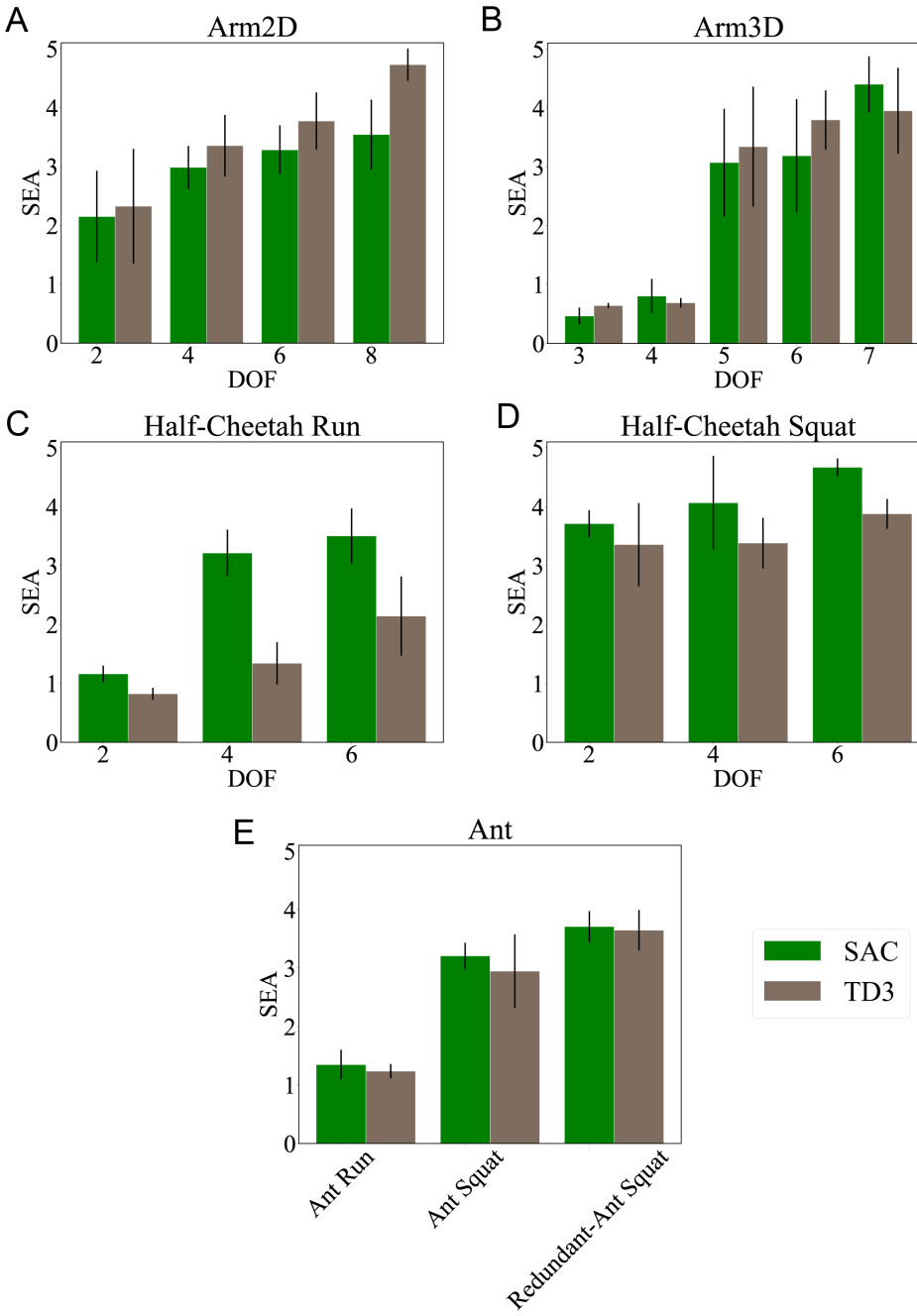


Figure 3.7: SEA metric evolution of SAC algorithm under different agent configurations. Standard deviation of the SEA is represented as shaded area around each curve. (A) SEA of the Arm2D agent during the training phase under different joint configurations. (B) SEA of the Arm3D agent during the training phase under different joint configurations. (C) SEA of the running Half-Cheetah agent during the training phase under different joint configurations. (D) SEA of the squatting Half-Cheetah agent during the training phase under different joint configurations. (E) SEA of the squatting Ant and Redundant-Ant agents during the training phase. (F) SEA of the running Ant agent during the training phase.

following task in real life. Indeed, humans can perform a circle drawing task with only the shoulder joints articulation together with the flexion of the elbow. As these four DOF are essentially required without having extra joint redundancy, the result (Fig. 3.7B) indicates that the SEA metric could capture well the redundancy situation of the arm given the joint configurations and the task. This result matches well with the known fact in robotics that four DOF is required for a circle drawing task, as a popular robotic design for this task is to use a 4-DOF arm robot. It is important to remark that no prior information on the robotic agent was provided to the DRL algorithms, and the SEA is obtained purely through the automated exploration process of the algorithms. However, the SEA metric does not allow for direct comparison of the joint redundancy between two different robotic agents, e.g., the Arm2D and Arm3D models, as they are structurally different and thus they do not share the same joint space.

In contrast, the results of the running and the squatting tasks for the Half-Cheetah (Fig. 3.7C, 4D) can be compared between them as they are based on the same robotic agent. The joint space of the less redundant version of the Half-Cheetah is a subset of that in the more redundant version in our experiments. It can be remarked that increasing the number of joints in the Half-Cheetah also results in an increase in the SEA value as the joint redundancy increases for the running task. However, the SEA values for the squatting task are higher than those for the running task regardless of the number of joints. This suggests that the current design of the Half-Cheetah agent is more suitable for the running task than the squatting task regardless of the joint configurations, as the joints are more fully exploited for the running task. Indeed, for the squatting task, the SEA does not increase significantly when more joints are added. This indicates that the added redundancy space is not relevant for this task. To better understand these two results, it must be noted that when the degree of joint redundancy is low, it indicates that the robotic agent does not possess the freedom to perform any secondary tasks, as all of the joints are engaged in executing the primary task [48][49], and vice versa. Using this concept, it is reasonable to conclude that the Half-Cheetah agent does not fully utilize its joint redundancy resource when executing the squatting task.

From the perspective of the DRL optimization process, higher SEA values indicate that significant exploration of the joint space is required before arriving at an acceptable solution. There are two factors that lead to this situation. In some cases, it is due to the DRL algorithm being indecisive between equally attractive solutions owing to the redundant joint space, which offers various approaches to solve the primary task. This suggests that there exists more than one way to solve the primary task, referred to as local optima in the DRL. For example, for the Half-Cheetah

running task (Fig. 3.7C), there are possibly a few suboptimal ways to perform running with six joints. The second factor that leads to a high SEA metric is an inappropriate choice of robotic agent for a task, which leads to a redundant joint space with the acceptable solution space concentrated in a small space. This is typically the case for the Half-Cheetah squatting task (Fig. 3.7D), as the robotic agent could possibly be jumping up and down instead of squatting, leading to extra exploration by DRL algorithms to find a solution and consequently a high SEA metric. In any case, when a higher SEA metric is observed relative to other configurations, there is indeed extra joint redundancy as explained above. The same results can be observed for the Ant experiments (Fig. 3.7E) with the same explanations as for the Half-Cheetah experiments. Interestingly, both the Ant and the Redundant-Ant configurations have the same number of joints, but the Redundant-Ant has a higher SEA value. This is because that the hip joints of the Redundant-Ant are configured to allow flexion to induce extra joint redundancy in the squatting task by allowing the Redundant-Ant to squat in a wider range. This property is accurately captured by the SEA metric.

The results between the SAC algorithm and the TD3 algorithm are fairly consistent in all cases. This is a positive sign that the proposed method is not specific for one particular DRL algorithm. The slight differences between these two algorithms, notably for the Half-Cheetah running task (Fig. 3.7C) and the Half-Cheetah squatting task (Fig. 3.7D), are due to the different exploration methods used by each algorithm during the training process. As the SEA values are higher in the case of the SAC algorithm, it can be concluded that the SAC algorithm explores more joint space than the TD3 algorithm. This result corresponds well to the known property of the SAC algorithm, which is its high exploration efficiency as it uses entropy maximization in its objective function [79]. This result also supports the idea that the SEA metric corresponds well to the exploration level of the solution space.

3.3.2 SEA for kinematics-specific and dynamics-specific joint redundancy quantification

In the previous subsection, it has been shown that the SEA metric can quantify the joint redundancy, as well as provide information that could never be easily obtained, such as the joint importance for a task and the task-specific joint redundancy. Indeed, it must be noted that the DOR is more complicated than just counting the number of joints that a robotic agent possesses. It is about the resources that an agent possess to achieve a task. The more DOR (resources) an agent has, the more ways there are to solve the given task. This is already well demonstrated through the

examples of the Arm3D results in Fig. 3.7B and also the task-specific joint redundancy results for the Half-Cheetah (Fig. 3.7C, D) and also the Ant (Fig. 3.7E). To further prove this point and demonstrate that the SEA metric can indeed quantify the DOR of an agent, two extra experiments are designed to verify if the SEA metric can quantify the joint redundancy affected by the kinematics and dynamics of an agent correctly. In the following, these two kinds of joint redundancy are named the kinematics-specific joint redundancy and the dynamics-specific joint redundancy. The 7-DOF Arm3D agent is used in these experiments as the human-like robotic arm can help one to easily understand the kinematics-specific and dynamics-specific joint redundancy by making analogies to one's daily life.

The kinematics-specific joint redundancy is affected by kinematic factors such as the range of motion of an agent. It affects the number of ways/ motions that an agent possesses to accomplish a task. For example, when a seated person wants to reach for a cup right in front of him, he has a lot of different ways to approach the cup as the cup is within his range of motion, and hence the kinematics-specific joint redundancy of his arm is high in this case. However, when the cup is put far enough from that person such that he can just barely touch the cup, he can only reach for the cup by extending his arm straight out to the direction of the cup, making the kinematics-specific joint redundancy of his arm low in this case. To show that the SEA metric can quantify the kinematics-specific joint redundancy as described in the previous examples, some experiments are carried out with the Arm3D agent. The previous circle-drawing task is reused here, but with the center of the circle being closer and also further from the Arm3D agent, hoping to recreate the scenario described previously. In Fig 3.8, there are three separate circle centers, i.e. C_1, C_2, and C_3, placed in front of the Arm3D agent one at a time for each experiment. The center C_2 is the same as the experiment done previously in Fig. 3.7B for the 7-DOF Arm3D agent, with the center being placed at a distance D from the origin of the coordinate space. The center C_1 is placed closer to the Arm3D agent, placed at a distance of $0.2D$ from the origin of the coordinate space. Finally, the center C_3 is placed further from the Arm3D agent at a distance of $2D$ from the origin of the coordinate space. The 7-DOF Arm3D agent is then required to draw circles around these centers. The C_3 case is designed such that the Arm3D agent can barely draw a circle (or a part of the circle) around the center C_3 as it is placed at the reaching limit of the agent. The C_1 case is then to further increase the kinematics-specific joint redundancy of the Arm3D agent. The experiments are carried out to verify if the SEA metric can quantify the kinematics-specific joint redundancy as expected. The sequences of the circle-drawing motion for C1, C2 and C3 cases can be found in the Fig. 3.9.

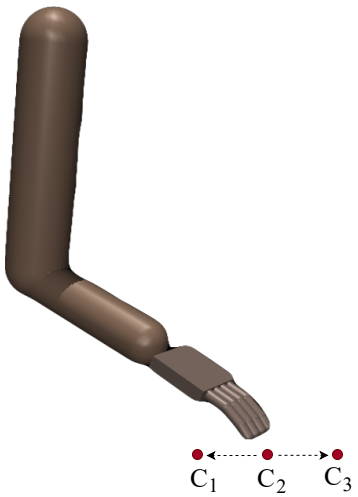


Figure 3.8: The 7-DOF Arm3D agent with different circle centers of the circle-drawing tasks. The center C_2 is the same as the one used in the experiment in the Fig. 3.7B, which is placed at a distance of D from the coordinate space origin. The center C_1 is placed at a distance of $0.2D$ from the coordinate space origin, while the center C_3 is placed at a distance of $2D$ from the coordinate space origin. The kinematics-specific joint redundancy of the Arm3D agent increases from the center C_1 to C_3.

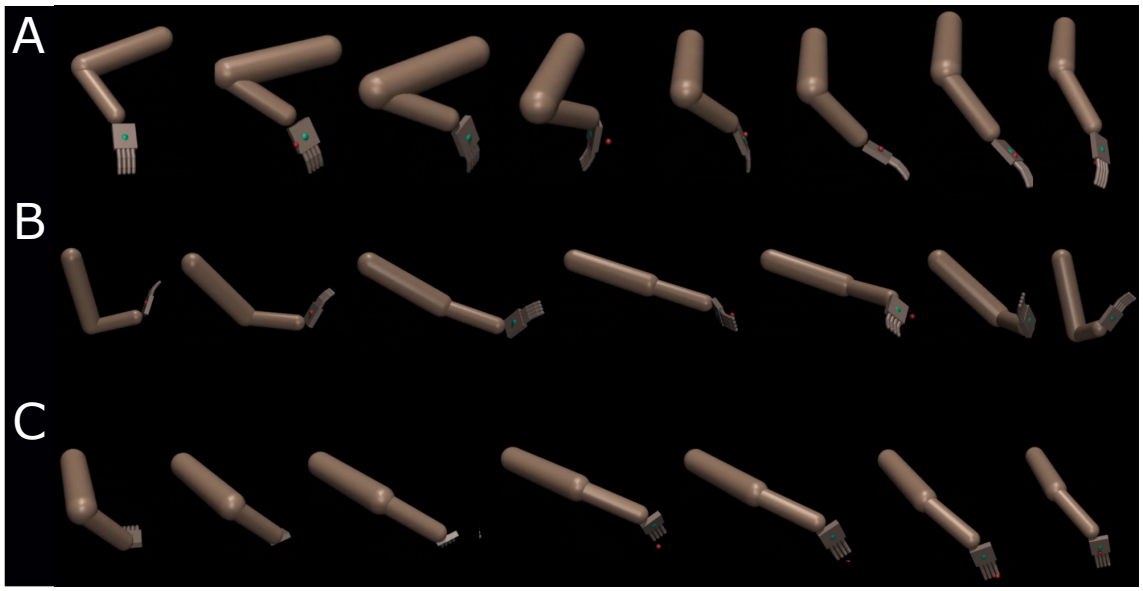


Figure 3.9: The circle-drawing motions for the centers (A) C_1, (B) C_2 and (C) C_3. For (A), it is obvious that the circle is within the reach of the Arm3D agent. For (C), the circle is almost out of reach of the Arm3D agent and the arm is almost always stretched out.

The experiment results can be found in the Fig. 3.10. From the results of the Fig. 3.10, it can be noticed that the SEA of the Arm3D agent decreases as the circle gets further from the agent. This verifies that the SEA metric can indeed quantify the kinematics-specific joint redundancy of the Arm3D agent. The SEA for the C_1 case and the C_2 case do not vary too much as the kinematics-specific joint redundancy to draw the circle is believed to be near its maximum value. Indeed, it is reasonable as all the circles which are close enough to the Arm3D agent will enable the agent to have an equally infinite ways to draw the circle, hence the upper limit on the kinematics-specific joint redundancy and also the SEA of the agent. As the circle gets almost out of reach from the Arm3D agent, the SEA metric decreases remarkably as the kinematics-specific joint redundancy also decreases in this case.

Next, it is also desirable that the SEA metric can quantify the dynamics-specific joint redundancy of an agent. The dynamics-specific joint redundancy is the redundancy that an agent possesses to accomplish a task subjected to its dynamic properties such as its weight and its force. For example, when a person holds a load on his hand, he can move his arm less freely than the case when he is not holding anything as the total weight on his arm has increased. This is a very simple and direct way to showcase the idea of dynamics-specific joint redundancy. A dif-

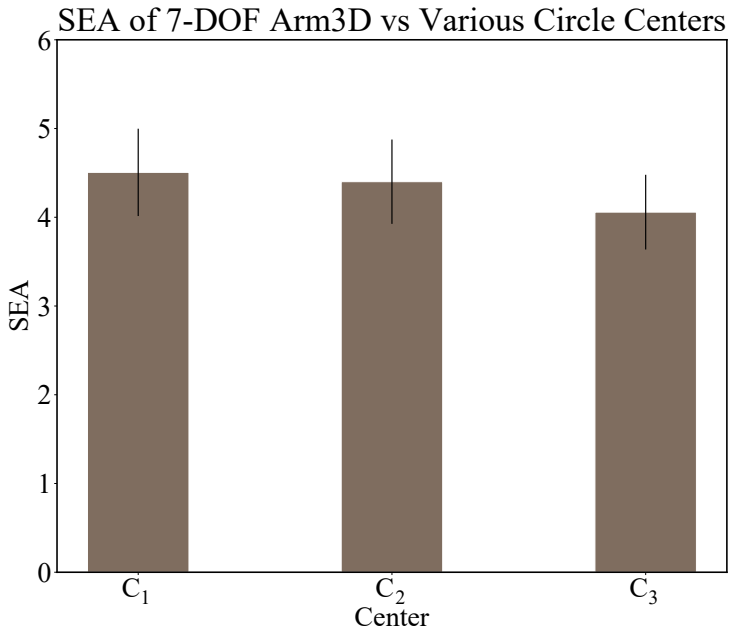


Figure 3.10: The bar plot of the SEA of the circle-drawing 7-DOF Arm3D agent as the circle center varies from C_1 to C_3. It can be seen that as the circle center gets further from the Arm3D agent (C_1 to C_3), the SEA metric decreases as well, verifying that the SEA metric can detect the decreasing kinematics-specific joint redundancy of the Arm3D agent in these experiments.

ferent example can be imagined by having the input force of the arm being lessen instead of adding a weight, both resulting to the same consequence of the reduced dynamics-specific joint redundancy. Some experiments are carried out to verify if the SEA metric can equally quantify the dynamics-specific joint redundancy of the Arm3D agent. In these experiments, the input torque of the Arm3D agent is varied for the same circle-drawing task. More specifically, if T is the input torque used in the experiments in Fig. 3.7B, two additional experiments are carried out for the 7-DOF Arm3D agent by changing the input torque to $0.25T$ and $0.5T$, i.e. reducing the dynamics-specific joint redundancy of the Arm3D agent. It is desired that the SEA metric can detect this change in dynamics-specific joint redundancy of the circle-drawing agent. The sequences of the circle-drawing motion for $0.25T$, $0.5T$ and T input torque cases can be found in the Fig. 3.11.

The experiment results are shown in the Fig. 3.12. It can be noticed that as the input torque of the Arm3D agent increases for the circle-drawing tasks, the SEA metric also increases accordingly. This demonstrates that the SEA metric is capable of quantifying the dynamics-specific joint redundancy as well. Indeed, as the input torque gets smaller, the Arm3D agent will have less options to draw the circles as some options are more torque-demanding. This result can also be reasoned by considering that the less input torque case corresponds to a heavier arm, and the more input torque case corresponds to a lighter arm. It is logical to do this analogy as when there is less input torque, it is more difficult to move the arm, therefore it is the same as if the arm being heavier but with a normal input torque. It is then possible to reason that as the Arm3D agent is heavier (has less input torque),

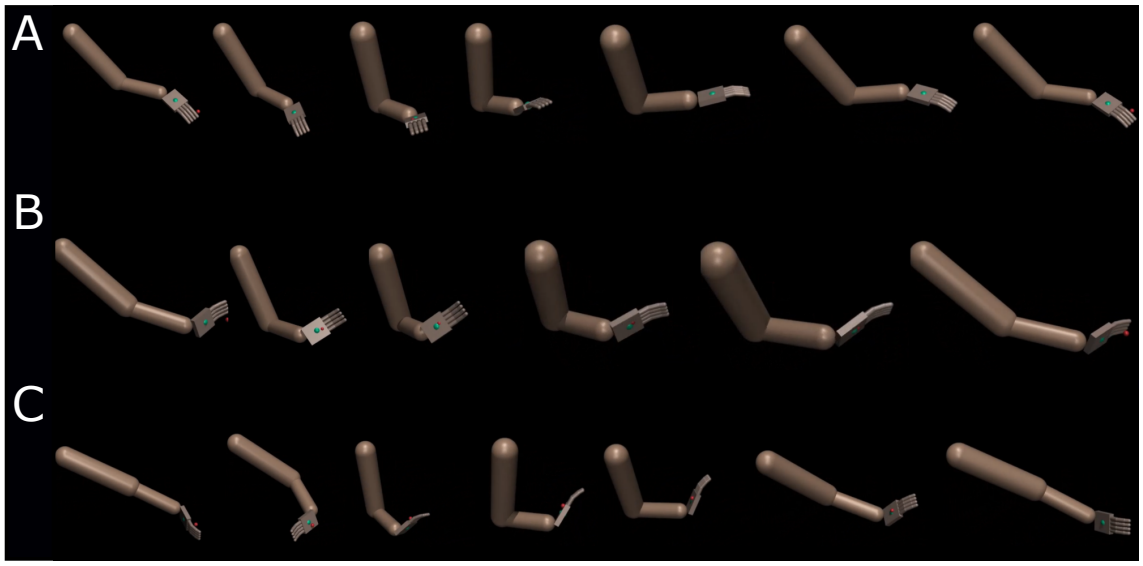


Figure 3.11: The circle-drawing motions for the input torque (A) $0.25T$, (B) $0.5T$ and (C) T. As the input torque increases from (A) to (C), it can be noticed that the arm can move more easily.

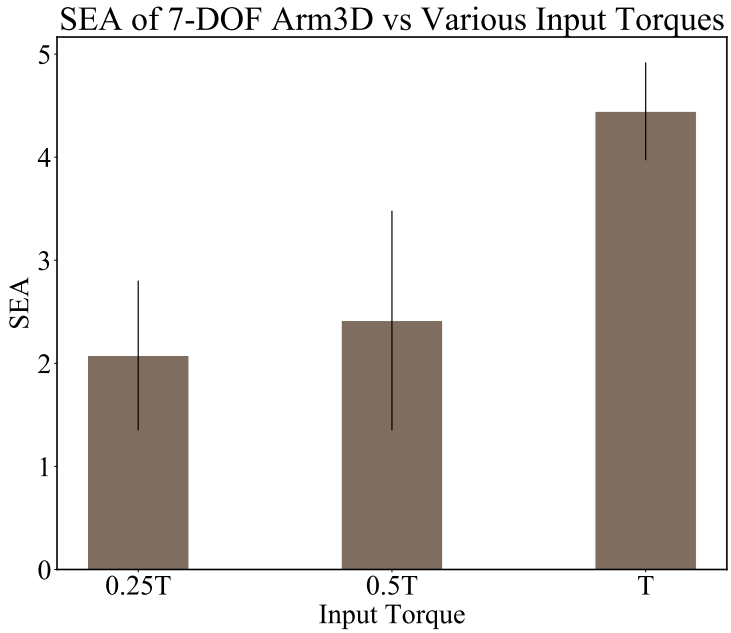


Figure 3.12: The bar plot of the SEA of the circle-drawing 7-DOF Arm3D agent as the input torque varies from $0.25T$ to T. It can be seen that as the input torque increases, the SEA metric increases as well, verifying that the SEA metric can detect the increasing dynamics-specific joint redundancy of the Arm3D agent in these experiments.

the dynamics-specific joint redundancy and the SEA decreases, and vice versa. The heavier arm (or less input torque) makes the agent has less ways to accomplish the circle drawing task (less dynamics-specific joint redundancy), and this is precisely captured by the SEA metric.

3.3.3 Application for body design evaluation

From the bar plots (Fig. 3.7), it is known that the SEA metric can provide extra information such as the importance of a certain joint for a given task. This leads to the intuition that the SEA metric can also be used to judge the optimality of a robotic structure for a given task, i.e., whether the assigned joint redundancy of a robotic structure is suitable for a task such that the DRL algorithm can find an optimal solution efficiently without excessive exploration. To this end, the plot of the task performance versus the SEA metric can be plotted to give us information about the relationship between the performance and the SEA metric. As shown in the Fig 3.13, the plot of the task performance versus the SEA metric can be divided into four different regions where each region gives certain information about the structural characteristics of the agents situated in this region. The top left corner of a plot

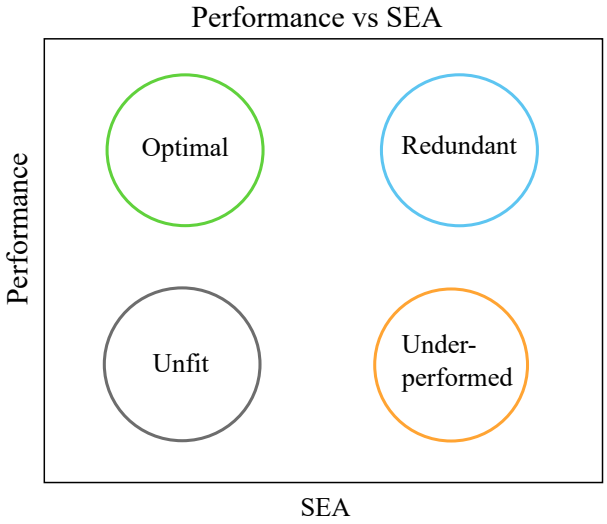


Figure 3.13: The plot of the task performance versus the SEA metric. The plot can be divided into four different regions, i.e. the optimal region (green) on the top left; the redundant region (blue) on the top right; the underperformed region (orange) on the bottom right; the unfit region (grey) on the bottom left. These regions give information about the structural characteristics of an agent for a certain task.

corresponds to the optimal structure region (green circle), as the DRL algorithm does not need to perform excessive exploration (small SEA value) to find a solution (high performance) to the task. The robotic agent is structurally optimal for the task without unnecessary joint redundancy. The top right region is associated with the redundant structure region (blue circle), as the DRL algorithm needs to explore significantly (high SEA value) to arrive at the solution (high performance) of the task. This shows evidence that the robotic agent is structurally redundant, and it could be used for other tasks as well. The bottom right region is the underperformed region (orange circle), as the excessive exploration of DRL algorithms (high SEA value) fails to lead to a good solution (low task performance). This indicates that the optimal solution is not reachable within the given robotic joint space despite excessive training, suggesting that the robotic agent is not suitable for the current task. Finally, the bottom left region (grey circle) corresponds to the unfit structure region, as the DRL algorithm fails to find any useful joint space (small SEA value) for the given task, leading to a low performance solution. This indicates that the robotic agent is simply inappropriately designed for the given task and it does not provide much possibility for further deployment.

With the above concept in mind, the plots of the task performance versus the SEA metric are plotted for each agent with the corresponding tasks (Fig. 3.14). From these plots, the aim is to gain insight into the effect of the DOF of an agent on the task performance. For the Arm2D results (Fig. 3.14A), the task performance, which is the tracking error between the fingertip of the agent and the target point

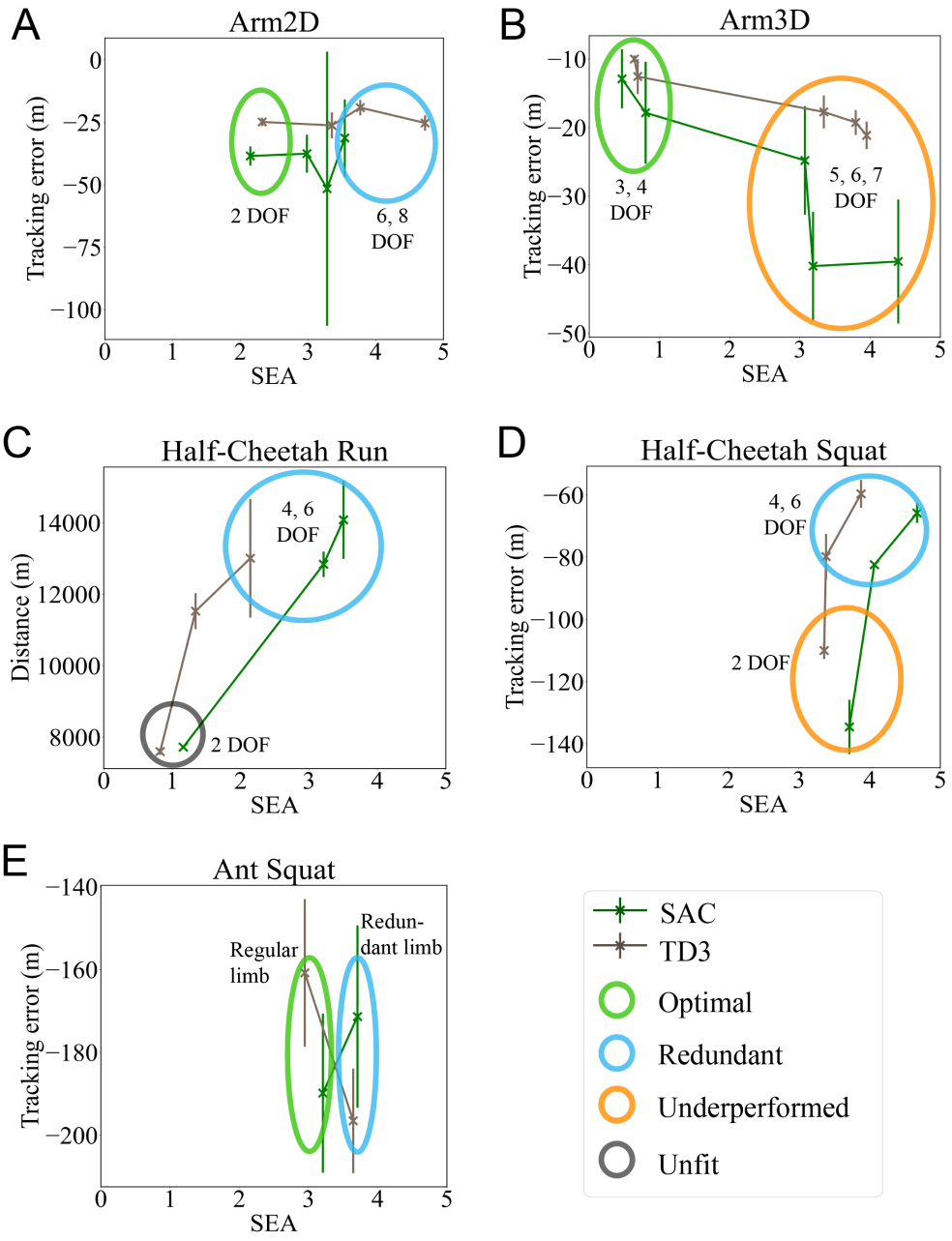


Figure 3.14: Task performance versus the SEA metric under different agent configurations. Each data point is associated with an agent having a certain DOF, with the DOF increases from the left side to the right side of each plot. Standard deviation of the task performance is represented by an error bar at each data point. Circles of different colors are the clusters of agents having the similar structural characteristics. (A) Target-tracking error of the Arm2D agent versus the SEA metric. (B) Target-tracking error of the Arm3D agent versus the SEA metric. (C) Total distance covered during the Half-Cheetah running task versus the SEA metric. (D) Target-tracking error with respect to the alternating high and low target points of the Half-Cheetah squatting task versus the SEA metric. (E) Target-tracking error with respect to the alternating high and low target points of the Ant squatting task versus the SEA metric. Performance of the Ant for the running task is not shown as there is just one data point

(Fig. 3.1A), does not vary too much as the DOF of the agent increases from two to eight. Hence, for the same task performance, the 2-DOF Arm2D agent has the optimal structure for the 2D planar tracking task, as little exploration (small SEA value) is required for finding a solution. This is not unexpected as we know that two DOF is sufficient for solving this task. The 6- and 8-DOF Arm2D agents are indeed redundant, indicated by the blue circle, as there is too much joint redundancy for this simple task. For the Arm3D agent (Fig. 3.14B), we can notice that there are two clearly distinct clusters of agents in the plot. The 3- and 4-DOF Arm3D agents cluster at the top left region of the plot, which is the optimal structure region, and the other Arm3D agents cluster at the bottom right region, which is the underperformed region. If the horizontal axis of Fig. 3.14B was the number of DOF, there would be no apparent cluster of agents in the plot. Indeed, the SEA metric has successfully grouped the agents into the respective optimal and underperformed regions, providing information about the optimal DOF of the Arm3D agent for the 3D circle drawing task. This example demonstrates that the SEA metric can guide the user to choose the optimal robotic structure for a task.

For the running (Fig. 3.14C) and squatting (Fig. 3.14D) Half-Cheetah results, the 4- and 6-DOF Half-Cheetah agents are in the redundant structure region (blue circle). These results support our interpretation of the plot that the agents situated in the redundant structure region are structurally redundant and they could be used for multiple different tasks. The 2-DOF Half-Cheetah agent achieves relatively low performance for both the running and squatting tasks in comparison to the other two configurations of the Half-Cheetah agent. It is not surprising that the 2-DOF agent is in the unfit region for the running task and the underperformed region for the squatting task as two DOF provides insufficient joint redundancy for executing complex locomotion skills. Through the example of this analysis, a real-life robotic agent having the same results as the 2-DOF Half-Cheetah agent would be eliminated from the choice of potential candidates for employment in future tasks. Finally, for the squatting Ant experiment (Fig. 3.14E), we can notice that the types of limb used (Fig. 2D) do not affect significantly the task performance. This result is similar to the Arm2D result (Fig. 3.14A) as the configurations of the agent do not significantly affect the task performance. In such cases, the agent with the lower SEA value is preferred as it restricts unnecessary joint redundancy for the task. It can be observed the SEA captures well the joint redundancy information for the given task in these results.

3.4 Discussion

One important aspect of the proposed joint redundancy quantification approach is that it is DRL-based. The particular feature of a DRL algorithm is the existence of a reward function, and the optimization process to find a solution for a robotic agent is guided by this predefined objective function. When a DRL-based approach is used to quantify the joint redundancy of a robotic agent, the quantification accuracy is also affected by the reward function. This is because the algorithm only explores the part of the joint space that seems promising based on the predefined reward function. The implication of this is that not all of the joint space is explored, and the SEA metric only quantifies the redundancy of the joint space in which a solution could exist according to the DRL reward function. While this may seem to be a limitation of the proposed approach, it could also be reasoned from a different perspective that this is actually an advantage. The primary objective of quantifying the joint redundancy is to determine the existence of a solution for a task in the considered joint space. The exclusion of the non-related joint space reduces the noise and increases the relevance of our approach for quantifying the joint redundancy of a robotic agent for a task.

Another issue is that it quantifies the joint redundancy of an agent relatively, i.e., the SEA value of an agent must be compared to another agent's SEA value to know their relative joint redundancy. The downside of this limitation is that one must train multiple agents under different configurations using DRL algorithms which may be time-consuming. Thus, one way to apply the proposed quantification approach in real-life robots is to first carry out a study in a simulation environment, and then transfer the acquired knowledge, e.g., the optimal robotic joint configuration for a task, to the real-life robots. Nevertheless, the utility of the proposed quantification approach is believed to be worth the training time, similar to the other state-of-the-art DRL methods [81][82]. The different regions of structural characteristics in Fig. 3.14 are also relatively defined by the existence of multiple clusters of data points of agents in the same plot. However, if the configurations of the agents in the experiments are sufficiently different among them, this limitation could be overcome easily as multiple clusters would exist consequently. Besides, the proposed methodology demands the use of high-performance DRL algorithms, e.g., the SAC algorithm [79] or the TD3 algorithm [80], as these algorithms allow the analysis to be focused on the joint redundancy aspects rather than the details of the DRL algorithm. Finally, as a consequence of using DRL algorithms, the proposed approach is inevitably subject to the randomness of the DRL optimization process; however, this does not seem to be a significant problem, as our results (Fig. 3.7) are

fairly consistent.

3.5 Conclusion

In this chapter, the deep reinforcement learning-based joint redundancy quantification approach is explained. Through the experimental results, it has been verified that the synergy exploration area (SEA) metric can convey information about the relative joint redundancy for various robotic agents and across different tasks. Indeed, the study has successfully applied the concepts of the DRL optimization process to the quantification of joint redundancy in robotic agents, and the SEA metric acts as a bridge between these two domains.

One might argue that the Degrees of Redundancy (DOR) can be obtained by simply subtracting the joint space dimension from the task space dimension. DOR is influenced by multiple factors such as the DOF, the joint structure design, the dynamics of the body (dynamics-specific joint redundancy), and the range of the motion related to the given task (kinematics-specific joint redundancy). In particular, for complex robots, such as high DOF cases or non-fixed base robots, it is not straightforward to apply the model-based approach since the agent-environment interaction cannot be fully modeled. By using the proposed DRL approach, one can make use of the exploration property of the DRL algorithm to obtain information not only about the DOR, but also additional information such as the importance of a joint for a given task, as shown by the example of the Arm3D agent (Fig. 3.7B), and the suitability of a robotic agent for a particular task, as shown by the performance analysis in Fig. 3.14. The SEA has also shown to be able to quantify the kinematics-specific redundancy (Fig. 3.10) and the dynamics-specific joint redundancy (Fig. 3.12) of a robotic agent, via the examples of the circle-drawing Arm3D agent. These types of information are not obtainable by subtracting the joint space dimension from the task space dimension or with analytical model-based quantification approaches [48][49][50][51][52]. This information can be used to select a better design for a robotic agent to ensure it has the appropriate degree of joint redundancy for an intended task, which will subsequently reduce the complexities of the final control method and prevent wasting resources by using an overly complex robot. This functionality of robotic design evaluation with the SEA metric can provide useful information to the robotics field, e.g., the computer-aided design for robotics. During the robotic design, the optimal joint redundancy to be added to a robot for a given task can be analyzed with consideration of the dynamic feasibility, as the solution exploration is performed under the physics-engine driven environment.

Chapter 4

Quadrupedal Energetic Analysis using Deep Reinforcement Learning

4.1 Introduction

In the previous two chapters, DRL has been used to train agents for synergetic analysis. More specifically, synergy-related metrics have been used to analyze the synergy properties of DRL-trained redundant agents. The role of DRL is passive in the previous two studies as it does not explicitly control the synergy properties of the trained-agents. In this chapter, DRL is used to actively manipulate the synergy properties of a quadruped robot, changing the gait mode of the robot as specified by the user. A gait mode specification method for DRL algorithms is introduced and its effectiveness in imposing a certain gait type is proven. The resulting trained quadruped robots are used in an energetic analysis to compare the energy efficiency between two gait modes, i.e. the gallop and trot modes. The joint-spring effect of the quadruped's joints are also varied to show the adaptability of a DRL algorithm to different dynamical properties of the quadruped. The proposed DRL approach would provide a framework for quadrupedal trot-gallop energetic analysis for different body structures, body mass distributions and joint characteristics with minimal hyperparameter tuning. Fig. 4.1 shows the DRL control loop of a quadruped robot used throughout this chapter.

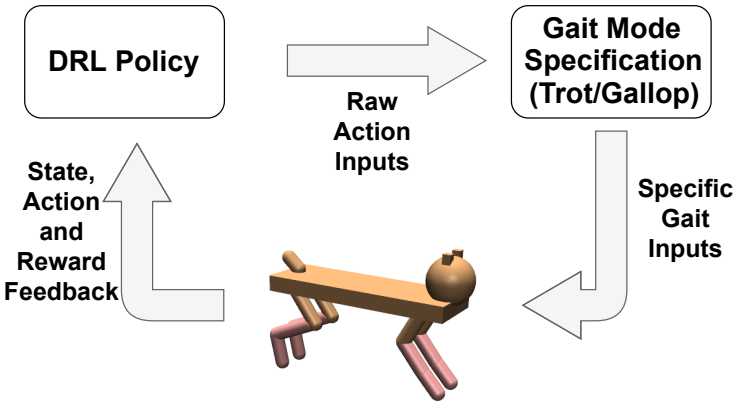


Figure 4.1: The control loop of a quadruped using deep reinforcement learning (DRL). The raw action inputs are transformed into specific gait inputs as specified by the user in the learning process, allowing the production of the desired gait mode locomotion.

4.2 Method

4.2.1 Simulated Quadruped Agent

In this paper, a quadruped model (Fig. 4.1) previously described in the previous chapters will be used. It is simulated with the Mujoco [67], a famous engine used in vaious DRL research [32][33] as it simulates realistic physical properties. As this study is the first step to validate the proposed idea, a simulated agent is sufficient as it is low-cost and allows fast experimentation. The agent has four limbs, with each limb having three joints, i.e. the hip joint, the knee joint and the foot joint. The stiffness and damping parameters of the joints can also be modified to simulate various degrees of passive joint-spring effect during the running motion.

4.2.2 DRL reward function and evaluation metrics

While it is possible to use any off-the-shelf DRL algorithms to carry out the experiments in this chapter, the state-of-the-art DRL algorithm, i.e. the SAC [32] algorithm is used. The only key characteristic of DRL concerned in this paper is the reward function employed during the learning process. For the experiments in this study, the robot is required to move forward in a two dimensional plane at a certain speed while considering at the same time its energy consumption issue. This can be translated to the reward function defined by the Equation (4.2.1):

$$R(t) = -|v(t) - v_{target}| - 0.1 \cdot \sum_i A_i(t)^2 \tag{4.2.1}$$

At each time step t, the algorithm has to minimize two terms in the reward function. The first term requires that the current speed of the quadruped robot, $v(t)$ matches as close as possible to a given target speed v_{target}. The second term of the reward function is a representation of the energy consumed by the quadruped agent, where $A_i(t)$ is the magnitude of the torque input for the joint i. This term is scaled by a small coefficient so that the algorithm will not converge to a sub-optimal solution of not moving at all.

In the following of this chapter, the performance metric is the distance travelled by the quadruped robot during a simulation of total time steps T, with each time step Δt being evaluated as one unit of time in the simulation environment. The performance metric can be written as the Equation (4.2.2):

$$Performance = \sum_{T} v(t) \cdot \Delta t \tag{4.2.2}$$

In this study, the energy expenditure index,which is the sum of the second terms of the reward function (4.2.1) throughout a running trajectory, can be written as the Equation (4.2.3):

$$Energy\ \ index = \sum_{T} \sum_{i} A_i(t)^2 \tag{4.2.3}$$

A third evaluation metric, called the performance-energy metric, is equally used in this study. The performance-energy metric is as described in the first chapter, with the energy term replaced by the energy index used in this study. The performance-energy metric can be written as the Equation 4.2.4. The performance-energy can be used to evaluate the energy-efficiency of a DRL-trained agent, considering both the performance and the energy expenditure of the agent at the same time.

$$Performance\text{-}energy = \frac{Performance}{Energy\ \ index} = \sum_{T} \frac{v(t) \cdot \Delta t}{\sum_i A_i(t)^2} \tag{4.2.4}$$

4.2.3 Gait Mode Specification

To specify a gait mode in the DRL learning process, the known fact of certain gait modes is exploited. In this study, only the gallop and the trot gaits are considered. For the gallop gait, it is known that the limbs on the right side of the quadruped

move symmetrically to the limbs on the left side. In reality, there might be a slight phase delay between the left and the right limbs in a gallop motion, but the assumption that there is no phase delay is made in this study. Therefore, the gallop gait is specified by having the DRL policy to produce torque inputs for the left limbs of the quadruped, then these torque inputs are copied identically for the right limbs, imposing the symmetric property in a gallop motion. This is translated as the Equation (4.2.5) where i and j are the corresponding joints on the right side and the left side of the quadruped respectively. The overview of the proposed control strategy is illustrated in the Fig. 4.2.

$$\tau_{right_i} = \tau_{left_j} \tag{4.2.5}$$

It must be noted that this does not constrain the DRL algorithm from finding an optimal solution as it can still freely output the torque for the left limbs while receiving feedback about the overall body kinematic condition. For the trot gait, the limbs on the right and the left side move asymmetrically to each other. Therefore, the DRL policy torque inputs for the left limbs are negated and copied to the right limbs to impose a trot gait, as described by the Equation (4.2.6):

$$\tau_{right_i} = -\tau_{left_j} \tag{4.2.6}$$

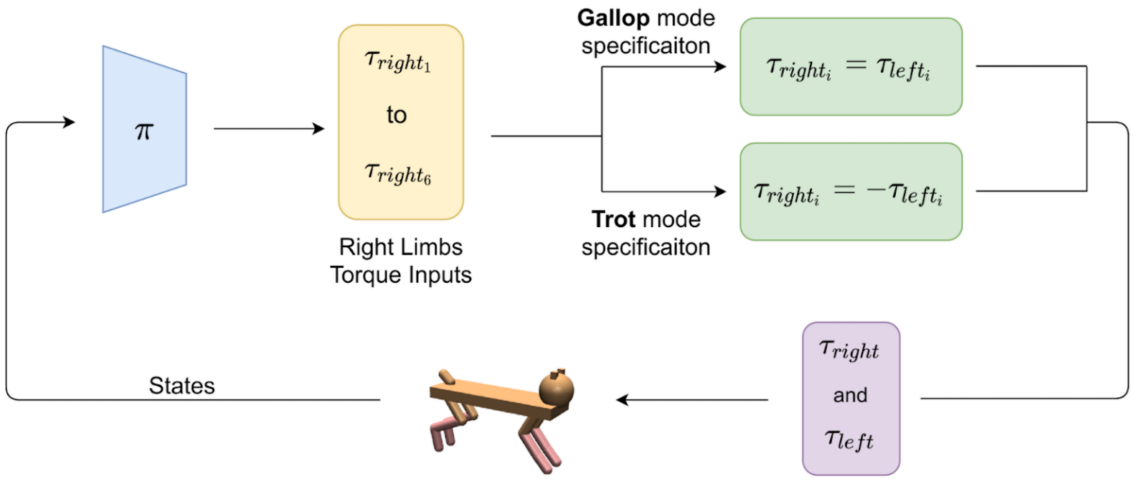


Figure 4.2: The DRL control loop with the gait mode specification methods detailed.

4.3 Experimental Results

For all the experiments, the quadruped robot is trained for 400 thousand time steps until convergence. Three trials of each experiment are conducted and the average results as well as the standard deviations are presented. The video for the quadruped locomotion can be found at `https://youtu.be/RD4Uvskp9Zg`.

4.3.1 Gait Mode Specification Effects

The effect of the gait mode specification on the DRL learning process is presented in this subsection. As illustrated on the left of the Figure 4.3, the performance of the quadruped robot with the trot and gallop mode specified converged faster than the case without any specification, showing that the gait mode specification has indeed sped up the DRL learning process. On the right of the Figure 4.3, it can be remarked that the energy consumption for all cases peaked near the beginning of the learning process as the algorithm was exploring a gait locomotion starting from random movements. This corresponds to the beginning phase of the performance graph on the left of the Figure 4.3 where the performance increased steeply. The energy consumption decreased steadily in the remaining of the training process as the DRL algorithm discovered a more energy efficient locomotion to move forward while spending less energy at the same time, as specified by the second term in the reward function (4.2.1). This clearly demonstrates the advantage of using DRL over a classical controller as the reward function can be easily tailored to take into account different aspects when carrying out a task, much like the learning process of living things. Figure 4.4 shows the performance-energy plot of the quadruped

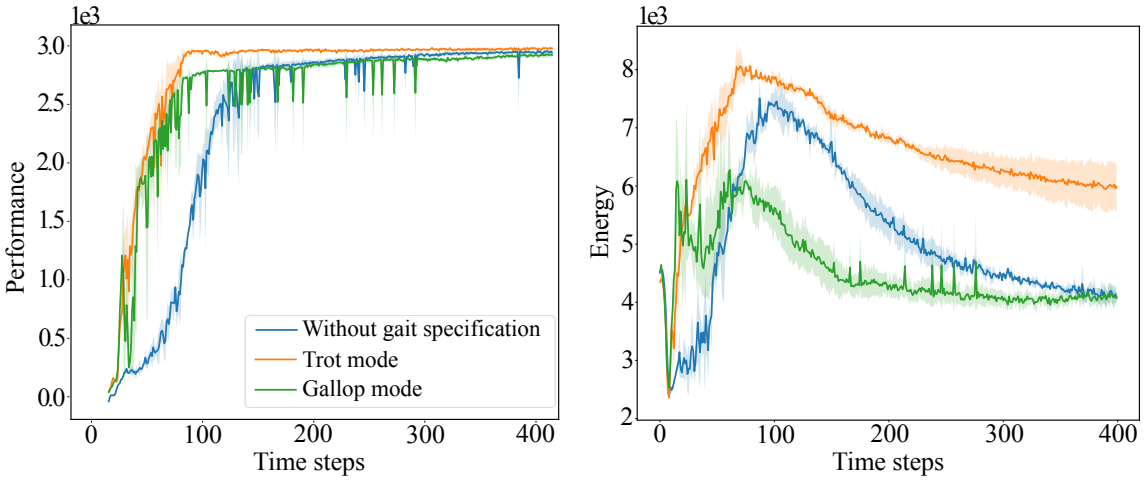


Figure 4.3: The performance (left) and energy metric (right) comparison throughout the training process between DRL-trained agents with no gait specification (blue), with a trot mode specification (orange) and with a gallop mode specification (green).

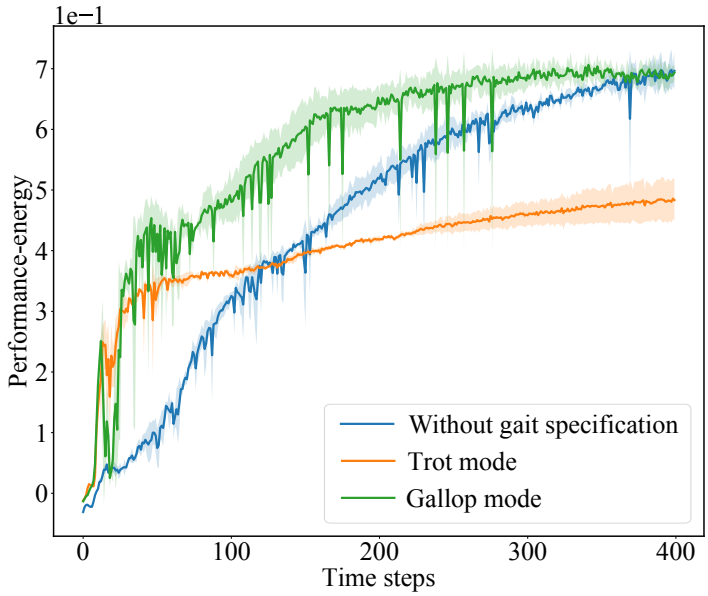


Figure 4.4: The performance-energy comparison throughout the training process between DRL-trained agents with no gait specification (blue), with a trot mode specification (orange) and with a gallop mode specification (green).

robots. This plot considers both the performance and the energy expenditure of the quadrupeds at the same time, enabling an overview of the energy efficiency of the quadrupeds with different gait modes. Due to the high performance and the low energy expenditure of the gallop mode, the performance-energy of the galloping quadruped is much better than the other cases as shown in the figure. On the other hand, the trotting quadruped is the least energy efficient due to its high energy expenditure. The quadruped without any gait mode specification lies between the other two cases, indicating that the agent without any gait mode specified might possess a gait which is a mix of the gallop and the trot gait, as it is demonstrated later.

Besides the faster convergence of the DRL learning process, the gait mode specification also successfully imposed a predetermined gait type on the quadruped robot. The Figure 4.5 shows the sequences of the quadrupedal gaits for different cases to give a visual idea on the motions. The gait diagram of each case is equally presented in the Figure 4.6. As shown on the left of the Figure 4.6, the gait diagram of the quadruped robot without any gait specification does not correspond to any known gait type. It is a mix between the gallop gait and the trot gait. Indeed, there is no guarantee that the output locomotion of a DRL-trained robot would possess the desired gait type, rendering the analysis done using a classical controller [28][27][59][24] impossible. However, by specifying a certain gait mode in the DRL training process using the proposed method, the output gait type resembles a well-known gait type, as shown by the gait diagram of the gallop mode and the trot mode in Figure 4.6.

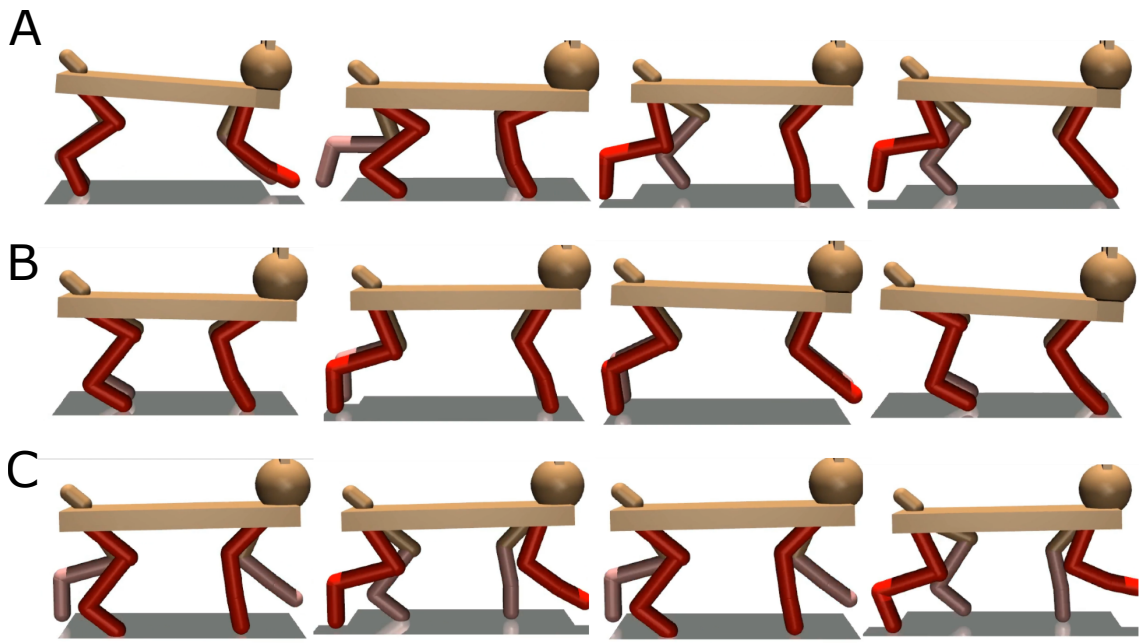


Figure 4.5: Sequences of the quadrupedal gaits after training. The right limbs are colored in red for better visualization. (A) The gaits of the quadruped without any mode specification. (B) The gaits of the quadruped with the gallop mode specified. (C) The gaits of the quadruped with the trot mode specified.

The effect of the gait mode specification can also be remarked in the spatial synergies of the quadruped while running with different gaits. The Figure 4.7 shows the spatial synergies of the quadruped with different gait mode specifications. It is expected that these spatial synergies will be able to provide information about the correlation between the joints of the quadruped while running. If there is a particular pattern inside the spatial synergy, then it indicates that there exists some form of joint coordination for running forward. It can be noticed that for the spatial synergy of the quadruped without any gait mode specification, there is no particular structure in the spatial synergy and all joints seem to be uncorrelated. This con-

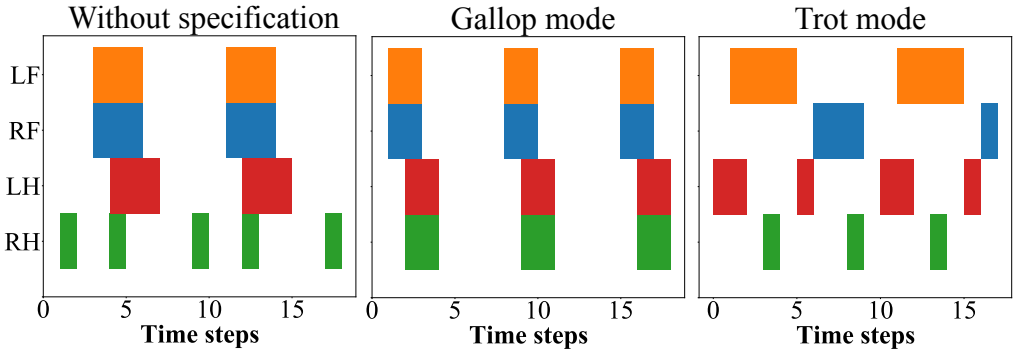


Figure 4.6: The gait diagram for quadruped robots with different gait mode specification. LF and RF represent the left fore limb and the right fore limb respectively, while LH and RH represent the left hind limb and the right hind limb respectively. The colored regions correspond to the stance phase of each limb.

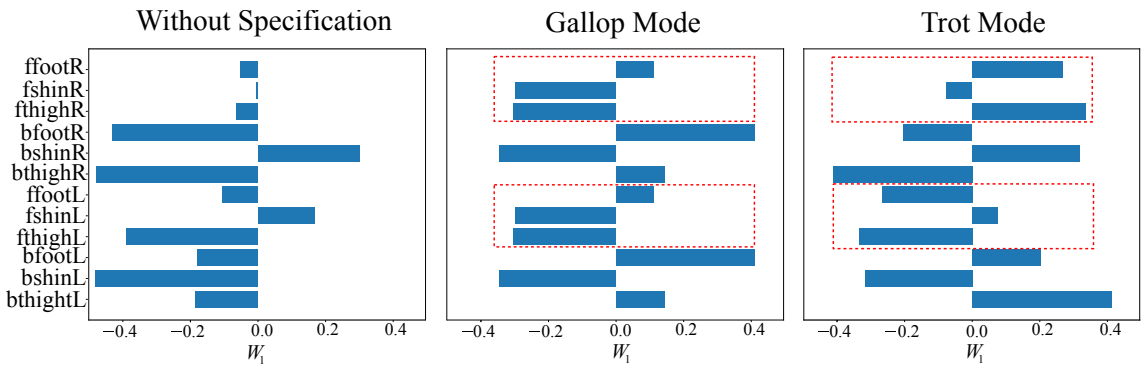


Figure 4.7: The spatial synergies of the quadruped with different gait mode specifications. Starting from the left is the spatial synergy of the quadruped without any specification; with the gallop mode specification (middle); with the trot mode specification (right). Only the most significant synergy is shown in this example. On the y-axis, the first letter of the label indicates whether it is a front limb (f) or a back limb (b); the last letter indicates whether it is a right limb (R) or a left limb (L). The red dotted boxes group the left and right front limbs together for easier interpretation.

firms that a DRL-trained quadruped does not necessary possess a well-known gait type if not specified explicitly. On the middle and the right plots of the Figure 4.7, it can be easily remarked that there exist some patterns in the spatial synergies of the quadruped when the gait mode is specified. The red dotted boxes group together the front limbs of the quadruped for easier interpretation. For the gallop mode specification, the left and right side of the limbs have the same values in the spatial synergy, indicating that both sides of the limbs are effectively coupled through our gait mode specification method, hence enabling the gallop motion through the synchronous motions of the fore limbs and the back limbs respectively. For the trot mode specification, the left and right side of the limbs have the same values but opposite signs in the spatial synergy, validating that our trot mode specification method has enforced the both side of the limbs to move in opposite phase, imitating the trot gait motion. In both spatial synergies of the two gait mode, the values of the fore and the back thigh joints are of opposite signs, indicating that they move constantly in opposite direction to move forward effectively.

The synergy development graph introduced in the first chapter of this thesis is also plotted for each quadruped with different gait modes, as shown in the Figure 4.8. In all cases, the synergy level increases throughout the training phase. This is as expected as the all the joints become more coordinated through training. However, there is a slight difference between the graphs of the quadruped with no specification and with specifications. It seems that the synergy level of the quadrupeds with gait mode specified converges faster. This may be due to the gait mode specifications which impose a certain constraint on the joint movements, hence making the synergy

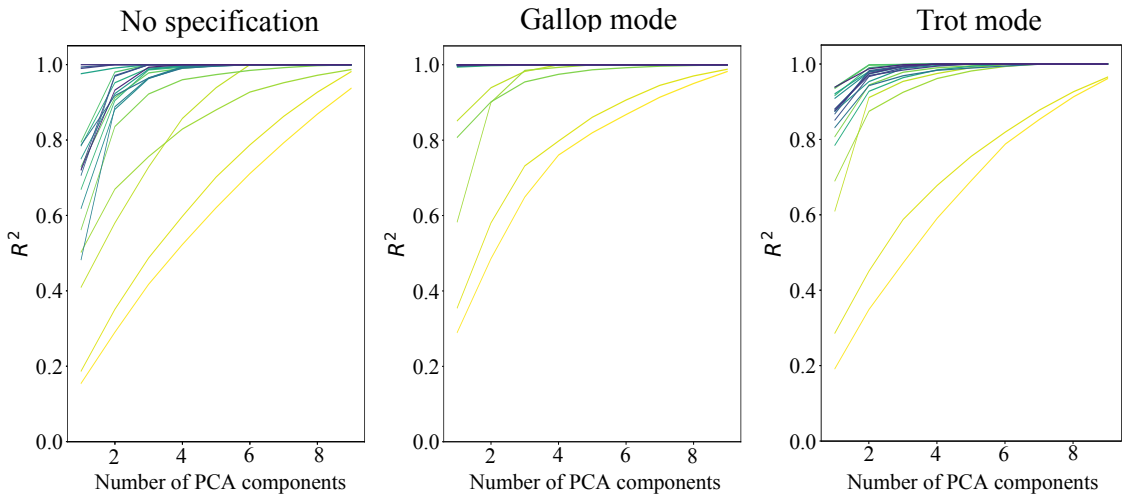


Figure 4.8: The synergy development graphs for quadrupeds with no gait specification (left); with the gallop mode specified (middle); with the trot mode specified (right). The color progression is as described in the first chapter, where the yellow color corresponds to the early phase of the training, and the purple color corresponds to the end phase of the training.

level to increase and converge faster. One plausible interpretation of this result is that the proposed gait mode specification during the learning process reduces the redundant solution space to a much smaller but predefined solution space, enabling the DRL algorithm to search for a solution in the reduced space much faster. This result is coherent with the performance graph in Figure 4.3, i.e. the gait mode specification promotes faster convergence in both the performance and the synergy level. This shows once again the benefit of introducing some user's knowledge in the DRL learning process, in this case the knowledge about certain gait modes.

4.3.2 Energetic Study between Gallop and Trot Gaits

Studies such as [28][27][24][60] suggest that certain gait types are more suitable for quadruped robots moving at certain speed. In particular, for moving at a higher velocity, the gallop gait is shown to be more energy efficient while for a lower walking speed, the trot gait is believed to be preferable. Motivated by this result, we have conducted a performance and energetic analysis on the quadrupedal gait motions generated by the DRL algorithm for two different target speeds, i.e. a target speed of 3 $m/\Delta t$ and a target speed of 5 $m/\Delta t$, where Δt is one time step of simulation. From the performance graph on the left of the Figure 4.9, for moving forward at a speed of 3 $m/\Delta t$, the performance of the trot gait is slightly higher than the performance of the gallop gait. However, for the moving speed of 5 $m/\Delta t$, the gallop gait is better than the trot gait. While the performance difference is small, this is still an encouraging result and shares a similarity with the previously established

results mentioned earlier. In term of the energy consumption during the forward motion, it can be observed on the right of the Figure 4.9 that the energy consumed by the galloping quadruped robots is always significantly lower than the trotting quadruped robots for both moving speeds. While this does not meet the expectation that the trot gait would consume less energy at a lower forward speed, one reason could be that the forward speed of 3 $m/\Delta t$ is not slow enough and the trot gait is a running trot gait in reality. It could also be that the passive joint-spring parameters of the quadruped robotic model favors the gallop gait in this study. This is reasonable as studies such as [83] has shown that the gallop gait is more energy efficient as it exploits the passive spring energy stored between the joints, helping the quadruped robot to move forward easily. This hypothesis may be supported by the Figure 4.10, which is the performance-energy plot between different gaits at different speeds. As it can be noticed easily in the Figure 4.10, the performance-energy of the galloping quadrupeds is always higher than the trotting quadrupeds at different speeds respectively, showing a higher energy efficiency of the galloping mode due to the exploitation of the passive spring energy between joints. However, as the current study is still at the early stage of a more complete study, the current result is promising as it shares some findings established in previous studies.

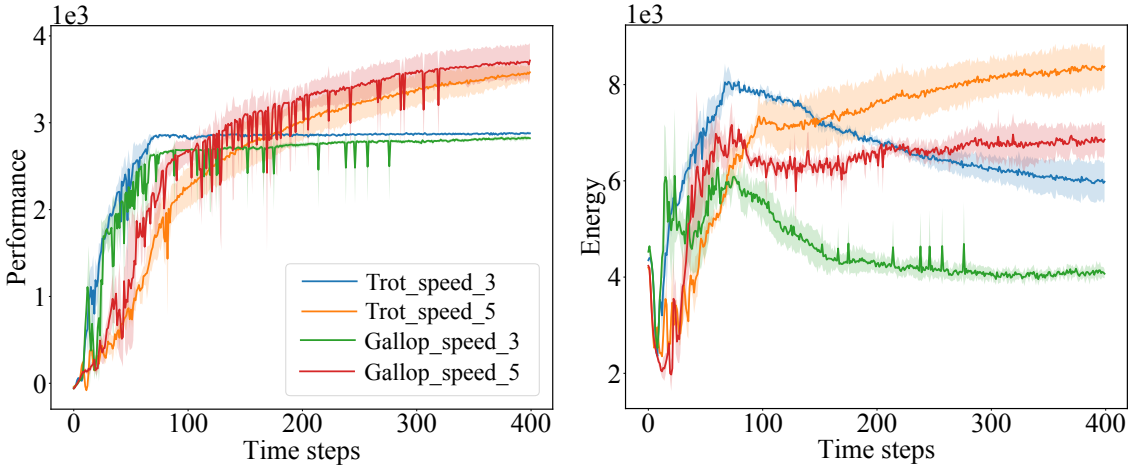


Figure 4.9: The performance (left) and energy metric (right) comparison between DRL-trained agents with a trot mode specification and a target speed of 3 (blue); with a trot mode specification and a target speed of 5 (orange); with a gallop mode specification and a target speed of 3 (green); with a gallop mode specification and a target speed of 5 (red). The random spikes on the curves are the deviations occurred during the DRL learning process.

4.3.3 DRL Exploitation of the Passive Joint-Spring Effect

In order to verify that the passive joint-spring effect plays an important role in the energy efficiency of the forward running motion of a quadruped as stated

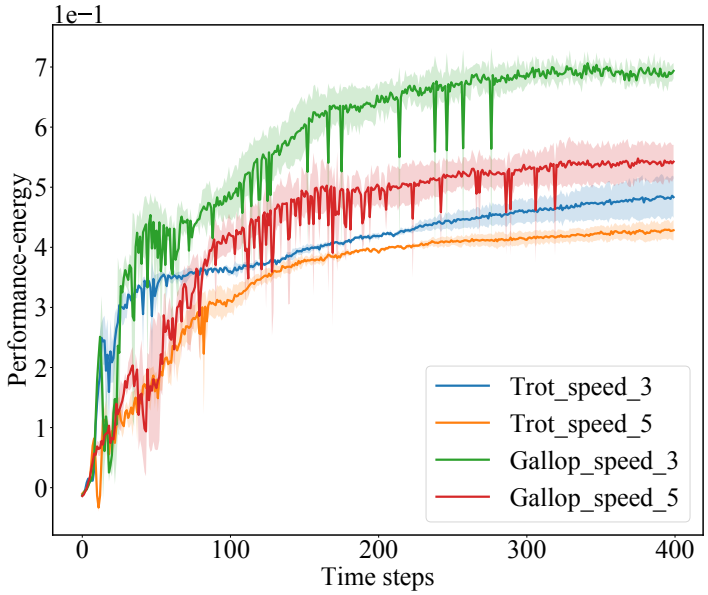


Figure 4.10: The performance-energy comparison throughout the training process between DRL-trained quadrupeds with different gait modes and different target running speeds.

in [83][84], the stiffness and the damping parameters of the joints of the galloping quadruped robot are modified from the default parameters used in the previous sections. To have a more joint-spring effect, the stiffness and the damping parameters are decreased for all the joints. On the other hand, to have a less joint-spring effect, the stiffness and the damping parameters are increased. As illustrated by the performance graph on the left of the Figure 4.11, the experimental results clearly show that the more joint-spring effect a quadruped robot has, the higher the galloping performance of the robot. In addition, from the energy curves on the right of the Figure 4.11, the energy consumption of the galloping motion also decreases as the joint-spring effect increases. This result matches perfectly with [83][84] as the passive joint-spring aids the running motion and reduces the energy consumed by the quadruped robot to gallop forward. The performance-energy plot in the Figure 4.12 demonstrates even clearer that the quadrupeds with more joint-spring effects are more energy efficient. This result also supports the idea that the DRL algorithm can serve as a general algorithm that can adapt to different physical conditions of a quadruped robot to produce gait motions for analysis, contrary to the case of classical controllers. Indeed, in the case of using classical controllers to generate gait motions, the parameters of the controllers need to be hand-tuned whenever there are changes in the physical properties of the quadruped robot or the experimental environment. DRL algorithms clearly have an advantage over classical controllers in this regard [64][65]. This result hence shows promises that DRL could provide framework for studies of quadrupeds with different physical and dynamical prop-

erties such as different weights, different joint properties, different structures, etc.

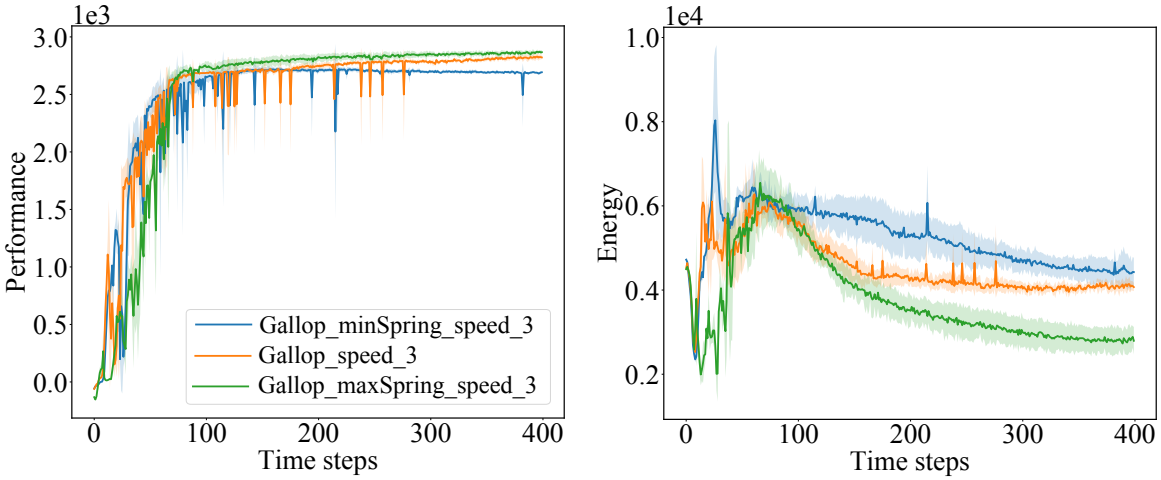


Figure 4.11: The performance (left) and energy metric (right) comparison between DRL-trained agents with varying joint-spring effects, i.e. the minimum joint-spring effect during running (blue), the default joint-spring effect (orange), and the maximum joint-spring effect during running (green). All the agents are with a gallop mode specification and a target speed of 3.

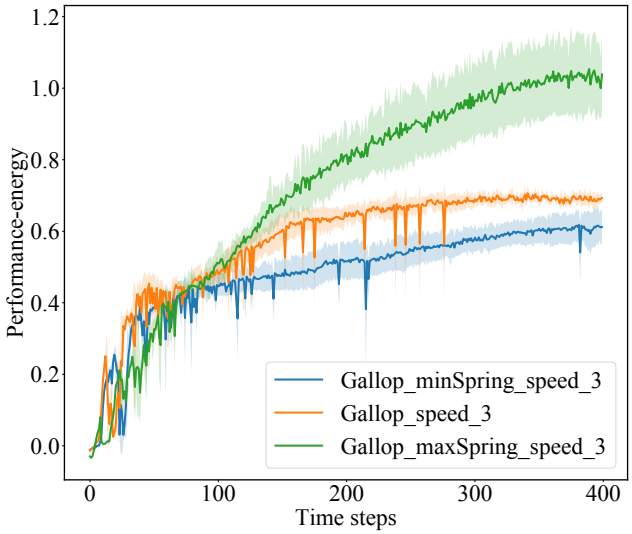


Figure 4.12: The performance-energy comparison throughout the training process between DRL-trained quadrupeds with the gallop mode specified by with different joint spring properties.

4.4 Discussion

From the experimental results, it has been shown that using DRL algorithms as an alternative to classical controllers for gait generations shows some promises for

quadruped energetic analysis. However, there are also several points that could be improved in the future.

Currently, the introduced gait mode specification works as expected to impose a certain gait type on the DRL-trained quadruped robot. As a continuation of this study, a more complicated gait mode specification method could be introduced to output a more precise gait locomotion. For example, as mentioned earlier that for a gallop gait, there might be a slight phase delay between the limbs on each side of the quadruped. A more complete gait mode specification method could possibly deal with this issue.

In the future, a more realistic quadruped model could be used for experiments. Ultimately, a real quadruped robot could be employed to carry out the same experiments in this paper to verify that the DRL-produced gait motion is plausible as well in a real robot. Ideally, different gait types could also be considered, so that the analysis done in works such as [28][27][24][60] can be repeated using DRL algorithms instead of classical controllers for gait generations.

4.5 Conclusion

In this chapter, it has been demonstrated that DRL algorithms show promises as an alternative to classical controllers for quadruped gait generations for energetic analysis. The specification of gait mode in the learning process speeds up the convergence of the algorithm as well as imposing a certain gait type on the quadruped robot, contrary to the case without any specification. DRL algorithms also show the ability to generalize to situations never seen before, providing optimal locomotion and removing the need for tedious manual parameters tuning in classical controllers. This work is the first step towards a more general framework of locomotion analysis in quadrupeds using DRL, contributing to the research field of understanding quadrupedal motion control on gait coordination.

Chapter 5

Conclusion

5.1 Summary

In this thesis, starting from the initial objective of recreating the synergy emergence process in simulated robotic agents using DRL, three resulting studies have been conducted. All the studies were inter-correlated and revolved around the motor synergy concept. Important results have been obtained from these studies, such as the relationship between the motor synergy and the energy efficiency of a motion, the quantification of joint redundancy using metric deriving from the motor synergy concept, and the effect of synergy mode specification on the convergence of the DRL learning process and the synergy emergence process. The DRL algorithms have also played an important role in this thesis as they were able to find a near optimal solution in a redundant solution space, allowing us to study the relationship between the discovery of an optimal solution and the synergy development process.

In chapter 2, in an attempt to understand the development of the correlation between robotic joints in the DRL learning process, i.e. the synergy emergence process, a synergy analysis which is inspired by the human motion researches was conducted. In the first part of the chapter, two state-of-the-art DRL algorithms were used to train several quadruped robots and their running motions were analyzed. The synergy development graphs were plotted and new synergy-related metrics were proposed in order to evaluate the synergy level of the robots during and after the DRL training phase. The experiment results showed that there exists a close relationship between the running performance, energy efficiency, and the synergy level of the robots throughout the training phase. More precisely, the higher the synergy level of the running robots, the higher the performance and energy efficiency of the robots. Although synergy-related constraint had never been encoded into the re-

ward function of the DRL algorithms, the synergy emergence phenomenon could be observed in all agents by purely optimizing the policy for the running performance only. This shows that there exists an inherent relation between the synergy level and the running performance, and this result shows certain similarities with some human studies as well as some robotic studies. It was also demonstrated that the SAC algorithm found solutions with higher performance and at the same time with higher synergy level than the TD3 algorithm, suggesting that a good DRL algorithm promotes high synergy level for better performance. This indicates that the proposed synergy metrics may be used for evaluating DRL algorithms in additional to classical performance metrics. In the second part of this chapter, in additional to the previous synergy analysis, a comparison was also made between the SAC algorithm and a classical PD controller in controlling a 7-DOF arm. The superiority of the DRL algorithm over the PD controller was demonstrated via the performance, energy efficiency and the synergy level of the 7-DOF arm in two tracking tasks.

In chapter 3, the focus was put on the quantification of the joint redundancy of some redundant robots using one of the previously proposed synergy-related metric, renamed to the SEA metric. The intuition behind the use of the SEA metric for the joint redundancy quantification was that the more redundant a robot is, the more exploration is needed for the DRL algorithm to find an optimal solution for a given task. Intuitively, the SEA metric which has been shown to be able to quantify the exploration of an DRL algorithm can thus potentially quantify the joint redundancy as well. Experiments were carried out in various redundant robots with various tasks, with the DRL algorithms used being both the SAC and the TD3 algorithms to show that the choice of the algorithm was not crucial in the proposed method. The experiment results showed that the SEA metric had successfully quantified relatively the joint redundancy of the robots with increasing complexity and with different tasks, showing promises to be better than analytical methods for joint redundancy quantification. Indeed, the analytical methods become intractable when the robots are too complex. Besides, the SEA metric was demonstrated to be able to provide additional useful information such as the importance of a certain joint for a certain task, and the difference in the redundancy of a robot when given different tasks. Experiments were equally carried to demonstrate that the SEA metric could equally quantify the kinematic and dynamic factors which affect the joint redundancy of a human-like robotic arm. At the end of the chapter, it was shown that it is possible to use the SEA metric for evaluating and choosing a suitable robotic design for a certain task in order to prevent overly complex robots being used for simpler tasks.

In chapter 4, several analysis were done on DRL-trained quadrupeds with different gait types explicitly specified before the experiments. The interest of this

study was to use the DRL algorithms for the gait generation in quadruped robots for various purposes. Different from the previous two chapters, the DRL algorithm was modified to actively modify the synergy properties of the robotic agents. In more details, gait specification methods were proposed in order to constrain the DRL algorithm to output a control policy that generates a certain gait type on the quadruped robots, as opposed to the case without any specification where there is no specific known gait type at the output. In this way, the DRL algorithm would modify the spatial synergy of the quadrupeds directly, hence imposing a certain synergy mode. The experiment results showed that with the proposed gait mode specification methods, the DRL learning phase and the synergy emergence process converged faster as the gait mode constraints reduce the redundant solution space into a smaller space. It was also demonstrated that the proposed gait mode specification methods had successfully imposed a certain gait type on the DRL-trained quadrupeds, in contrary to the case where there was no specification. An energetic study had also been carried out with the DRL-trained quadrupeds to compare the performance and energy efficiency of the quadrupeds with the gallop and the trot gaits, and with different target forward speeds. This is a typical study being carried out on quadrupeds, and it is known that the gallop gait is better suited for moving at a higher speed. With the DRL-generated gaits in this chapter, similar results awerere reproduced, i.e. the performance of the galloping quadruped is higher than the trotting quadruped at a higher target running speed, and the opposite for a lower target running speed. This showed promises that the DRL algorithm could indeed be used in quadrupedal studies. To further show the advantage of a DRL algorithm, the passive joint spring property of the galloping quadruped robots was modified. The DRL algorithm had shown to be successfully adapted to the changed dynamics of the quadrupeds, and was able to exploit the passive joint spring effect when galloping. This adaptive properties of the DRL algorithm is clearly superior to the classical controllers which need to be readjusted when the properties of the robots are changed. The results of this chapter demonstrated that the DRL algorithm shows promises to be able to provide a framework for studying quadruped motion with different physical and dynamical properties.

5.2 Contribution

The main contributions of this thesis are the following.

- Through the recreation of the synergy emergence process using DRL, the close relationship between the performance, energy efficiency and the synergy level

of a moving DRL-trained robotic agents was demonstrated.

- The proposition of new synergy-related metrics for analyzing robotic motions and also DRL algorithms.
- The proposition of a DRL-based relative joint redundancy quantification method validated through various simulated robots of different complexities and different given tasks.
- The proposition of gait specification methods in DRL algorithms for quadrupedal studies.
- The demonstration of the possibility to use DRL algorithms as an alternative to classical controllers, as well as the advantages of the DRL algorithms over the classical controllers in certain cases.

5.3 Future Work

There are several improvements and further investigations that can be done to the studies carried out in this thesis.

First of all, all the studies in this thesis were based on simulated robotic agents. While the MuJoCo simulation engine can simulate physics to a high degree of accuracy, however there is still a gap between the simulation environments and the real world. One possible future work is hence to reproduce the results in this thesis by conducting studies on real world robots. This should be a very challenging task as the DRL algorithms are known to be difficult to train in real world situation as they need a lot of training iterations, in addition to the wear and tear done to the robots as time passes. Another possible improvement that could be done to the study done in the first chapter is to solve the weaknesses of the synergy calculation method. Indeed, for the current synergy calculation method, a static agent will have high synergy level as the zero signal is easily reconstructed. Hence, further research is needed to overcome this weakness. It is also mentioned in the chapter 2 that only the agents which are stable in its static state are used as it makes the synergy calculation easier. However, it is preferable to be able to carry out the synergy analysis on wider range of robotic agents and hence, a better synergy calculation method is needed to take into account the non-stable robotic agents such as a biped. It is also desirable in the future to design an DRL algorithm which explicitly promotes the synergy level as it might improve the learning ability of the DRL algorithm.

Besides, more comparisons between the results of DRL-based methods and clas-

sical methods also need to be carried out in the future. For example, the DRL-based redundancy quantification method proposed in the chapter 3 can be compared to the results of some analytical methods. In the chapter 4, the analysis results of the DRL-generated quadrupedal gaits should also be compared to classical controllers-generated gaits. The gait mode specification methods introduced in the chapter 4 can also be improved to generate more accurate gait outputs. Indeed, currently, it is assumed that for the gallop gait, there is no phase delay between the left limbs and the right limbs of the quadruped robots. This is not accurate as there exists a small phase delay between the left and right limbs in a galloping quadruped.

List of Publications

J. Chai and M. Hayashibe, "Motor Synergy Development in High-Performing Deep Reinforcement Learning Algorithms," in *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 1271-1278, April 2020.

J. Chai and M. Hayashibe, "Quantification of Joint Redundancy considering Dynamic Feasibility using Deep Reinforcement Learning," in ICRA 2021.

J. Chai and M. Hayashibe, "Deep Reinforcement Learning with Gait Mode Specification for Quadrupedal Trot-Gallop Energetic Analysis," in EMBC 2021.

J. Han, J. Chai and M. Hayashibe, "Synergy Emergence in Deep Reinforcement Learning for Full-dimensional Arm Manipulation," in *IEEE Transactions on Medical Robotics and Bionics*vol. 3, no. 2, pp. 498-509, May 2021.

J. Han, J. Chai and M. Hayashibe, "Emergence of Motor Synergy in Multi-directional Reaching with Deep Reinforcement Learning," *2021 IEEE/SICE International Symposium on System Integration (SII)*, 2021, pp. 78-82.

Bibliography

[1] I. Delis, P. M. Hilt, T. Pozzo, S. Panzeri, and B. Berret, "Deciphering the functional role of spatial and temporal muscle synergies in whole-body movements," *Scientific Reports*, vol. 8, no. 1, p. 8391, May 2018.

[2] R. E. Singh, K. Iqbal, G. White, and T. E. Hutchinson, "A systematic review on muscle synergies: From building blocks of motor behavior to a neurorehabilitation tool," *Applied Bionics and Biomechanics*, vol. 2018, p. 3 615 368, Apr. 2018.

[3] R. Gentner, S. Gorges, D. Weise, K. Aufm Kampe, M. Buttmann, and J. Classen, "Encoding of motor skill in the corticomuscular system of musicians," *Current biology : CB*, vol. 20, pp. 1869–74, Oct. 2010.

[4] N. Dominici, Y. P. Ivanenko, G. Cappellini, A. d'Avella, V. Mondìà, M. Cicchese, A. Fabiano, T. Silei, A. Di Paolo, C. Giannini, R. E. Poppele, and F. Lacquaniti, "Locomotor primitives in newborn babies and their development," *Science*, vol. 334, no. 6058, pp. 997–999, 2011.

[5] E. Bizzi and V. C. Cheung, "The neural origin of muscle synergies," *Frontiers in Computational Neuroscience*, vol. 7, p. 51, 2013.

[6] F. T. Zaal, K. Daigle, G. L. Gottlieb, and E. Thelen, "An unlearned principle for controlling natural movements," *Journal of Neurophysiology*, vol. 82, no. 1, pp. 255–259, 1999.

[7] A. d'Avella, A. Portone, L. Fernandez, and F. Lacquaniti, "Control of fast-reaching movements by muscle synergy combinations," *Journal of Neuroscience*, vol. 26, no. 30, pp. 7791–7810, 2006.

[8] K. Arulkumaran, M. P. Deisenroth, M. Brundage, and A. A. Bharath, "Deep reinforcement learning: A brief survey," *IEEE Signal Processing Magazine*, vol. 34, no. 6, pp. 26–38, Nov. 2017.

[9] L. Tai and M. Liu, "Deep-learning in mobile robotics - from perception to control systems: A survey on why and why not," *CoRR*, vol. abs/1612.07139, 2016. arXiv: `1612.07139`.

[10] J. Lee, J. Hwangbo, L. Wellhausen, V. Koltun, and M. Hutter, "Learning Quadrupedal Locomotion over Challenging Terrain," *arXiv e-prints*, arXiv:2010.11251, arXiv:2010.11251, Oct. 2020. arXiv: 2010.11251 [cs.RO].

[11] V. Tsounis, M. Alge, J. Lee, F. Farshidian, and M. Hutter, "Deepgait: Planning and control of quadrupedal gaits using deep reinforcement learning," *IEEE Robotics and Automation Letters*, vol. PP, pp. 1–1, Mar. 2020.

[12] J. O'Doherty, J. Cockburn, and W. Pauli, "Learning, reward, and decision making," *Annual Review of Psychology*, vol. 68, Jan. 2017. DOI: 10.1146/annurev-psych-010416-044216.

[13] G. E. Wimmer, J. K. Li, K. J. Gorgolewski, and R. A. Poldrack, "Reward learning over weeks versus minutes increases the neural representation of value in the human brain," *Journal of Neuroscience*, vol. 38, no. 35, pp. 7649–7666, 2018, ISSN: 0270-6474. DOI: 10.1523/JNEUROSCI.0075-18.2018. eprint: https://www.jneurosci.org/content/38/35/7649.full.pdf. [Online]. Available: https://www.jneurosci.org/content/38/35/7649.

[14] M. Banich and S. Floresco, "Reward systems, cognition, and emotion: Introduction to the special issue," *Cognitive, Affective, and Behavioral Neuroscience*, vol. 19, May 2019. DOI: 10.3758/s13415-019-00725-z.

[15] J. P. Bhanji and M. R. Delgado, "The social brain and reward: Social information processing in the human striatum," *WIREs Cognitive Science*, vol. 5, no. 1, pp. 61–73, 2014. DOI: https://doi.org/10.1002/wcs.1266. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/wcs.1266. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/wcs.1266.

[16] E. Chiovetto, A. d'Avella, and M. Giese, "A Unifying Framework for the Identification of Motor Primitives," *arXiv e-prints*, Mar. 2016. arXiv: 1603.06879.

[17] F. M. Ramos, A. d'Avella, and M. Hayashibe, "Identification of time-varying and time-scalable synergies from continuous electromyographic patterns," *IEEE Robotics and Automation Letters*, vol. 4, no. 3, pp. 3053–3058, Jul. 2019.

[18] S. A. Overduin, A. d'Avella, J. Roh, J. M. Carmena, and E. Bizzi, "Representation of muscle synergies in the primate brain," *Journal of Neuroscience*, vol. 35, no. 37, pp. 12 615–12 624, 2015.

[19] A. Scano, L. Dardari, F. Molteni, H. Giberti, L. M. Tosatti, and A. d'Avella, "A comprehensive spatial mapping of muscle synergies in highly variable upper-limb movements of healthy subjects," *Frontiers in Physiology*, vol. 10, p. 1231, 2019.

[20] F. Fahimi, "Autonomous robots: Modeling, path planning, and control," 2008.

[21] M. Prats, P. J. Sanz, and A. P. del Pobil, "The advantages of exploiting grasp redundancy in robotic manipulation," in *The 5th International Conference on Automation, Robotics and Applications*, 2011, pp. 334–339.

[22] P. Singh, S. Jana, A. Ghosal, and A. Murthy, "Exploration of joint redundancy but not task space variability facilitates supervised motor learning," *Proceedings of the National Academy of Sciences*, vol. 113, no. 50, pp. 14 414–14 419, 2016.

[23] A. Ghosal, "Resolution of redundancy in robots and in a human arm," *Mechanism and Machine Theory*, Jan. 2018.

[24] D. Owaki and A. Ishiguro, "A quadruped robot exhibiting spontaneous gait transitions from walking to trotting to galloping," *Scientific Reports*, vol. 7, Dec. 2017.

[25] S. Seok, A. Wang, M. Y. Chuah, D. Otten, J. Lang, and S. Kim, "Design principles for highly efficient quadrupeds and implementation on the MIT cheetah robot," in *2013 IEEE International Conference on Robotics and Automation, Karlsruhe, Germany, May 6-10, 2013*, IEEE, 2013, pp. 3307–3312.

[26] Wikipedia contributors, *Gait — Wikipedia, the free encyclopedia*, [Online; accessed 3-April-2021], 2021. [Online]. Available: `https://en.wikipedia.org/w/index.php?title=Gait&oldid=1004889619`.

[27] M. Silva and J. Tenreiro Machado, "Energy efficiency of quadruped gaits," in. Feb. 2006, pp. 735–742.

[28] K. Kiguchi, Y. Kusumoto, K. Watanabe, K. Izumi, and T. Fukuda, "Energy-optimal gait analysis of quadruped robots," *Artificial Life and Robotics*, vol. 6, pp. 120–125, Sep. 2002.

[29] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*, Second. The MIT Press, 2018.

[30] K. Arulkumaran, M. P. Deisenroth, M. Brundage, and A. A. Bharath, "Deep reinforcement learning: A brief survey," *IEEE Signal Processing Magazine*, vol. 34, no. 6, pp. 26–38, 2017.

[31] Z. Bing, C. Lemke, Z. Jiang, K. Huang, and A. Knoll, *Energy-efficient slithering gait exploration for a snake-like robot based on reinforcement learning*, 2019. arXiv: `1904.07788 [cs.RO]`.

[32] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," in *Proceedings of the 35th International Conference on Machine Learning*, vol. 80, PMLR, Jul. 2018, pp. 1861–1870.

[33] S. Fujimoto, H. van Hoof, and D. Meger, *Addressing function approximation error in actor-critic methods*, 2018. arXiv: 1802.09477 [cs.AI].

[34] S. Shaharudin, D. Zanotto, and S. Agrawal, "Muscle synergy during wingate anaerobic rowing test of collegiate rowers and untrained subjects," *International Journal of Sports Science*, vol. 4, pp. 165–172, Aug. 2014.

[35] J. M. Inouye and F. J. Valero-Cuevas, "Muscle synergies heavily influence the neural control of arm endpoint stiffness and energy consumption," *PLOS Computational Biology*, vol. 12, no. 2, pp. 1–24, Feb. 2016.

[36] M. Chhabra and R. A. Jacobs, "Properties of Synergies Arising from a Theory of Optimal Motor Behavior," *Neural Computation*, vol. 18, no. 10, pp. 2320–2342, Oct. 2006, ISSN: 0899-7667. DOI: 10.1162/neco.2006.18.10.2320. eprint: https://direct.mit.edu/neco/article-pdf/18/10/2320/816599/neco.2006.18.10.2320.pdf. [Online]. Available: https://doi.org/10.1162/neco.2006.18.10.2320.

[37] M. T. Rosenstein, A. G. Barto, and R. E. Van Emmerik, "Learning at the level of synergies for a robot weightlifter," *Robotics and Autonomous Systems*, vol. 54, no. 8, pp. 706–717, 2006, Morphology, Control and Passive Dynamics, ISSN: 0921-8890. DOI: https://doi.org/10.1016/j.robot.2006.03.002. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0921889006000698.

[38] S. Bitzer, M. Howard, and S. Vijayakumar, "Using dimensionality reduction to exploit constraints in reinforcement learning," in *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Oct. 2010, pp. 3219–3225.

[39] K. S. Luck, G. Neumann, E. Berger, J. Peters, and H. B. Amor, "Latent space policy search for robotics," in *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Sep. 2014, pp. 1434–1440.

[40] K. S. Luck, J. Pajarinen, E. Berger, V. Kyrki, and H. B. Amor, "Sparse latent space policy search," in *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, Phoenix, Arizona, 2016, pp. 1911–1918.

[41] Z. Bing, C. Lemke, Z. Jiang, K. Huang, and A. Knoll, "Energy-efficient slithering gait exploration for a snake-like robot based on reinforcement learning," *CoRR*, vol. abs/1904.07788, 2019. arXiv: 1904.07788.

[42] J. Kober, J. A. Bagnell, and J. Peters, "Reinforcement learning in robotics: A survey," *International Journal of Robotics Research*, vol. 32, no. 11, pp. 1238–1274, 2013.

[43] R. Dubey and J. Luh, "Redundant robot control for higher flexibility," in *Proceedings. 1987 IEEE International Conference on Robotics and Automation*, vol. 4, 1987, pp. 1066–1072.

[44] E. Guigon, P. Baraduc, and M. Desmurget, "Computational motor control: Redundancy and invariance," *Journal of Neurophysiology*, vol. 97, no. 1, pp. 331–347, 2007.

[45] G. Palli and C. Melchiorri, "On the control of redundant robots with variable stiffness actuation," in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2012, pp. 5077–5082.

[46] C. Salaün, "Learning models to control redundancy in robotics," Aug. 2010.

[47] H. Sadeghian, V. Luigi, M. Keshmiri, and B. Siciliano, "Multi-priority control in redundant robotic systems," vol. 31, Sep. 2011, pp. 3752–3757.

[48] J. Lenarcic, "On the quantification of robot redundancy," in *Proceedings 1999 IEEE International Conference on Robotics and Automation*, vol. 4, 1999, 3159–3164 vol.4.

[49] ——, "The range of self-motion of redundant robots," in *1999 IEEE/ASME International Conference on Advanced Intelligent Mechatronics*, 1999, pp. 386–391.

[50] Fan-Tien Cheng, Chung-Wen Chen, and Min-Hsiung Hung, "Redundancy indices of kinematically redundant manipulators," in *2000 26th Annual Conference of the IEEE Industrial Electronics Society. IECON 2000. 2000 IEEE International Conference on Industrial Electronics, Control and Instrumentation. 21st Century Technologies*, vol. 1, 2000, 572–577 vol.1.

[51] Min-Hsiung Hung, Fan-Tien Cheng, and Jen-Kuei Ting, "A novel quantitative measure of redundancy for kinematically redundant manipulators," in *Proceedings 2001 ICRA. IEEE International Conference on Robotics and Automation*, vol. 4, 2001, 4060–4065 vol.4.

[52] Y. Nakamura, *Advanced Robotics: Redundancy and Optimization*, 1st. USA: Addison-Wesley Longman Publishing Co., Inc., 1990, ISBN: 0201151987.

[53] P. Singh, S. Jana, A. Ghosal, and A. Murthy, "Exploration of joint redundancy but not task space variability facilitates supervised motor learning," *Proceedings of the National Academy of Sciences*, vol. 113, no. 50, pp. 14 414–14 419, 2016, ISSN: 0027-8424. DOI: 10.1073/pnas.1613383113. eprint: https://www.pnas.org/content/113/50/14414.full.pdf. [Online]. Available: https://www.pnas.org/content/113/50/14414.

[54] H. K. Chu and M. Hayashibe, "Discovering interpretable dynamics by sparsity promotion on energy and the lagrangian," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 2154–2160, 2020.

[55] J. Chai and M. Hayashibe, "Motor synergy development in high-performing deep reinforcement learning algorithms," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 1271–1278, 2020.

[56] J. Baillieul and T. Samad, *Encyclopedia of Systems and Control*. Springer Publishing Company, Incorporated, 2015.

[57] Z. Hasan and J. Thomas, "Chapter 33 kinematic redundancy," in *Peripheral and Spinal Mechanisms in the Neural Control of Movement*, ser. Progress in Brain Research, M. Binder, Ed., vol. 123, Elsevier, 1999, pp. 379–387. DOI: https://doi.org/10.1016/S0079-6123(08)62872-1. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0079612308628721.

[58] J.-F. Yang, J. Scholz, and M. Latash, "The role of kinematic redundancy in adaptation of reaching," *Experimental Brain Research*, vol. 176, pp. 54–69, 2006.

[59] K. Yang, X. Rong, L. Zhou, and Y. Li, "Modeling and analysis on energy consumption of hydraulic quadruped robot for optimal trot motion control," *Applied Sciences*, vol. 9, p. 1771, Apr. 2019.

[60] P. Arena, L. Patanè, and S. Taffara, "Energy efficiency of a quadruped robot with neuro-inspired control in complex environments," *Energies*, vol. 14, no. 2, 2021.

[61] H. Chai, X. Rong, X. Tang, and Y. Li, "Gait-based quadruped robot planar hopping control with energy planning," *International Journal of Advanced Robotic Systems*, vol. 13, no. 1, p. 20, 2016.

[62] Y. Fukuoka, Y. Habu, and T. Fukui, "A simple rule for quadrupedal gait generation determined by leg loading feedback: A modeling study," *Scientific reports*, vol. 5, p. 8169, Feb. 2015.

[63] J. Chai and M. Hayashibe, "Motor synergy development in high-performing deep reinforcement learning algorithms," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 1271–1278, 2020.

[64] Honglak Lee, Yirong Shen, Chih-Han Yu, G. Singh, and A. Y. Ng, "Quadruped robot obstacle negotiation via reinforcement learning," in *Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006.*, 2006, pp. 3003–3010.

[65] G. Bellegarda and Q. Nguyen, "Robust Quadruped Jumping via Deep Reinforcement Learning," *arXiv e-prints*, arXiv:2011.07089, arXiv:2011.07089, Nov. 2020.

[66] B. Hu, S. Shao, Z. Cao, Q. Xiao, Q. Li, and C. Ma, "Learning a faster locomotion gait for a quadruped robot with model-free deep reinforcement learning," in *2019 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, 2019, pp. 1097–1102.

[67] E. Todorov, T. Erez, and Y. Tassa, "Mujoco: A physics engine for model-based control," *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 5026–5033, 2012.

[68] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba, *Openai gym*, 2016. eprint: `arXiv:1606.01540`.

[69] M. Hayashibe and S. Shimoda, "Synergetic motor control paradigm for optimizing energy efficiency of multijoint reaching via tacit learning," *Frontiers in Computational Neuroscience*, vol. 8, p. 21, 2014.

[70] T. Haarnoja, A. Zhou, K. Hartikainen, G. Tucker, S. Ha, J. Tan, V. Kumar, H. Zhu, A. Gupta, P. Abbeel, and S. Levine, "Soft actor-critic algorithms and applications," *CoRR*, vol. abs/1812.05905, 2018. arXiv: `1812.05905`. [Online]. Available: `http://arxiv.org/abs/1812.05905`.

[71] A. d'Avella and F. Lacquaniti, "Control of reaching movements by muscle synergy combinations," *Frontiers in computational neuroscience*, vol. 7, p. 42, 2013.

[72] F. Lacquaniti, G. Ferrigno, A. Pedotti, J. Soechting, and C. Terzuolo, "Changes in spatial scale in drawing and handwriting: Kinematic contributions by proximal and distal joints," *Journal of Neuroscience*, vol. 7, no. 3, pp. 819–828, 1987.

[73] Y.-w. Tseng and J. P. Scholz, "Unilateral vs. bilateral coordination of circle-drawing tasks," *Acta psychologica*, vol. 120, no. 2, pp. 172–198, 2005.

[74] T. Krabben, B. I. Molier, A. Houwink, J. S. Rietman, J. H. Buurke, and G. B. Prange, "Circle drawing as evaluative movement task in stroke rehabilitation: An explorative study," *Journal of neuroengineering and rehabilitation*, vol. 8, no. 1, p. 15, 2011.

[75] M. Hayashibe and S. Shimoda, "Synergetic learning control paradigm for redundant robot to enhance error-energy index," *IEEE Transactions on Cognitive and Developmental Systems*, vol. 10, no. 3, pp. 573–584, Sep. 2018, ISSN: 2379-8920.

[76] L. Dipietro, H. Krebs, S. Fasoli, B. Volpe, J. Stein, C. Bever, and N. Hogan, "Changing motor synergies in chronic stroke," *Journal of neurophysiology*, vol. 98, pp. 757–68, Sep. 2007. DOI: 10.1152/jn.01295.2006.

[77] S. Bhattacharyya, S. Shimoda, and M. Hayashibe, "A synergetic brain-machine interfacing paradigm for multi-dof robot control," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 46, pp. 1–12, Jul. 2016.

[78] E. Todorov, T. Erez, and Y. Tassa, "Mujoco: A physics engine for model-based control," *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 5026–5033, 2012.

[79] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," in *Proceedings of the 35th International Conference on Machine Learning*, vol. 80, PMLR, Jul. 2018, pp. 1861–1870.

[80] S. Fujimoto, H. van Hoof, and D. Meger, "Addressing function approximation error in actor-critic methods," *CoRR*, vol. abs/1802.09477, 2018.

[81] K. Arulkumaran, M. P. Deisenroth, M. Brundage, and A. A. Bharath, "Deep reinforcement learning: A brief survey," *IEEE Signal Processing Magazine*, vol. 34, no. 6, pp. 26–38, 2017.

[82] L. Tai and M. Liu, "Deep-learning in mobile robotics - from perception to control systems: A survey on why and why not," *CoRR*, vol. abs/1612.07139, 2016.

[83] N. C. Holt, T. J. Roberts, and G. N. Askew, "The energetic benefits of tendon springs in running: Is the reduction of muscle work important?" *Journal of Experimental Biology*, vol. 217, no. 24, pp. 4365–4371, 2014.

[84] A. Sutrisno and D. J. Braun, "How to run 50% faster without external energy," *Science Advances*, vol. 6, no. 13, 2020.

Acknowledgements

I want to express my gratitude to my advisor Professor Mitsuhiro Hayashibe, for providing me with the opportunity to work at the Neuro-Robotics Lab, and for his continuous support and guidance during my doctoral degree.

I would also like to thank my thesis committee members, Professor Mitsuhiro Hayashibe, Professor Koichi Hashimoto, Professor Satoshi Murata, and Dr. Juan Alvaro Gallego, for their thoughtful questions and valuable comments regarding this research project.

Additionally, I am thankful for Professor Dai Owaki's insightful comments on the last chapter of the thesis.

I am equally grateful to all my colleagues and members of the Neuro-Robotics Lab.

Finally, I want to thank my family for supporting me to complete this doctoral degree.