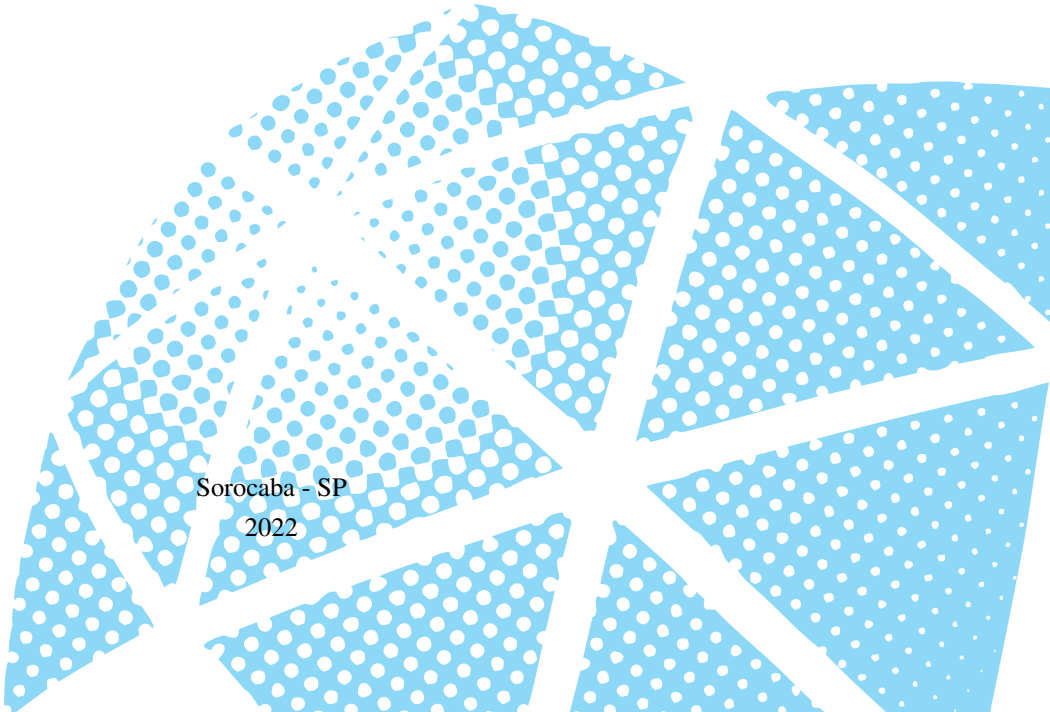
UNIVERSIDADE ESTADUAL PAULISTA
"JÚLIO DE MESQUITA FILHO"
Instituto de Ciência e Tecnologia de Sorocaba

NAYARI MARIE LESSA

Transferring human movements from videos to robots with Deep Reinforcement Learning



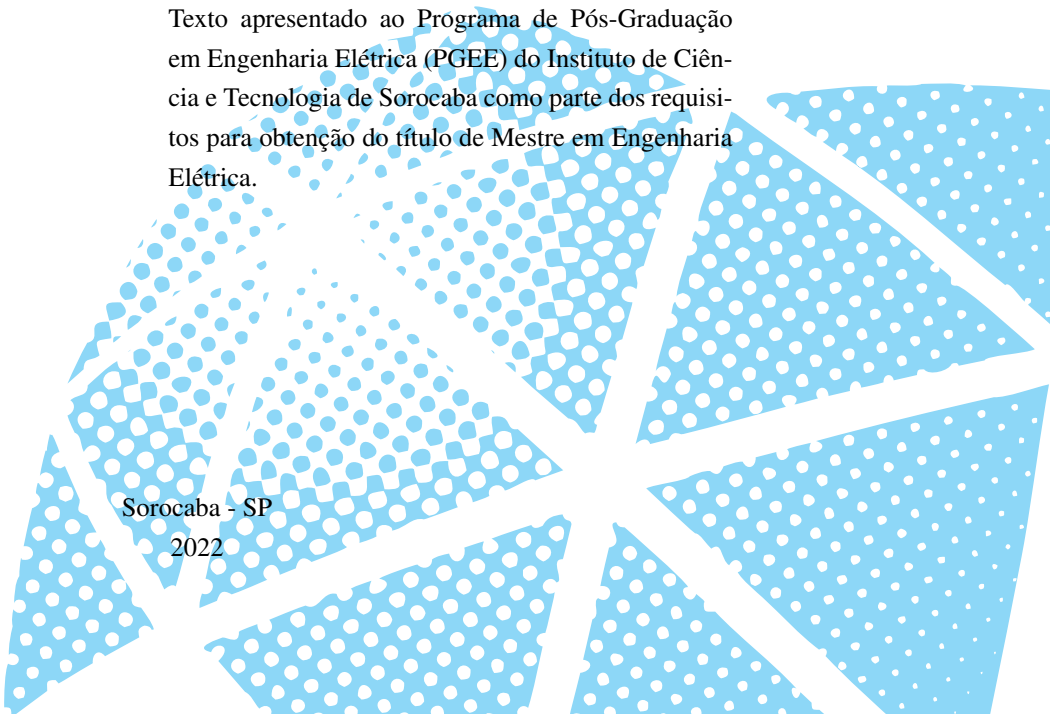
Sorocaba - SP
2022

NAYARI MARIE LESSA

Transferring human movements from videos to robots with Deep Reinforcement Learning

Transferindo movimentos humanos de videos para robôs com Aprendizado por Reforço Profundo

Texto apresentado ao Programa de Pós-Graduação em Engenharia Elétrica (PGEE) do Instituto de Ciência e Tecnologia de Sorocaba como parte dos requisitos para obtenção do título de Mestre em Engenharia Elétrica.



Sorocaba - SP
2022

L638t

Lessa, Nayari Marie
Transferring human movements from videos to robots with Deep Reinforcement Learning / Nayari Marie Lessa. -- Sorocaba, 2022
102 p.

Dissertação (mestrado) - Universidade Estadual Paulista (Unesp), Instituto de Ciência e Tecnologia, Sorocaba
Orientador: Alexandre da Silva Simões
Coorientadora: Esther Luna Colombini

1. Artificial intelligence. 2. Imitation. 3. Robots Dynamics. 4. Motion. I. Título.

Sistema de geração automática de fichas catalográficas da Unesp. Biblioteca do Instituto de Ciência e Tecnologia, Sorocaba. Dados fornecidos pelo autor(a).

Essa ficha não pode ser modificada.

UNIVERSIDADE ESTADUAL PAULISTA

Câmpus de São João da Boa Vista

CERTIFICADO DE APROVAÇÃO

TÍTULO DA DISSERTAÇÃO: Transfering human movements from videos to robots with Deep Reinforcement Learning. Transferindo movimentos humanos de vídeos para robôs com Aprendizado por Reforço Profundo

AUTORA: NAYARI MARIE LESSA
ORIENTADOR: ALEXANDRE DA SILVA SIMÕES
COORIENTADOR: ESTHER LUNA COLOMBINI

Aprovada como parte das exigências para obtenção do Título de Mestra em ENGENHARIA ELÉTRICA, área: Automação pela Comissão Examinadora:

Prof. Dr. ALEXANDRE DA SILVA SIMÕES (Participaçao Virtual)
Departamento de Engenharia de Controle e Automação / Instituto de Ciência e Tecnologia - UNESP - Câmpus de Sorocaba

Profª. Drª. ANNA HELENA REALI COSTA (Participaçao Virtual)
Engenharia de Computação e Sistemas Digitais / Universidade de São Paulo

Prof.ª Dr.ª PAULA DORNHOFER PARO COSTA (Participaçao Virtual)
Depto. de Engenharia de Computação e Automação Industrial / Faculdade de Engenharia Elétrica e de Computação (FEEC)- UNICAMP

São João da Boa Vista, 02 de junho de 2022

Maria Luiza Sarubi Barreto
Diretora Técnica Acadêmica

Acknowledgements

I thank all my family, friends, teachers and employees of the Institute of Science and Technology (ICT), who contributed directly or indirectly to the realization of this work. In particular, I would like to thank:

- My mother Maria Auxiliadora who has always been present, giving me support and love;
- My father Aparecido Donizeti who leaving me free in all my decisions, and for the support;
- Andre for your love, friendship, and encouragement;
- My cousin Danilo for all the conversations, friendship, and for the support;
- Prof. Dr. Alexandre and Dra. Esther, for all the teaching, encouragement, trust and guidance;
- My friends and colleagues who helped me directly or indirectly, especially Maria Fernanda and Renata Maria, for the help and work done together;
- GitHub community for democratizing access to the source code of advances and discoveries, which contributes significantly to the continued development of technology and science for innovation;
- The Federal Institute of São Paulo - Campus Salto for the incentive;
- This study was partially financed by the *Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil* (CAPES).

"Do the difficult things while they are easy and
do the great things while they are small.
A journey of a thousand miles must begin
with a single step"
Lao Tzu

Abstract

The study of humanoid robots in the field of robotics has grown in recent decades in the direction of developing robots able to support humans in many applications. The evolution of machine learning techniques, particularly the Reinforcement Learning (RL) approach, expanded the robotics domains to many new applications, based on the strategy to reinforce the agent through its interactions with the environment. Deep Reinforcement Learning (DRL) came to improve the RL technique allowing the application of robotics in highly complex task and scenarios. However, this approach is well known for two major disadvantages: i) its high computational cost; ii) the difficulty in training the robot to achieving particular policies that are usually very difficult to model. Recently, RL approaches based on the imitation of reference movements have emerged in the robotics scenario. The learning process in this approach is based on the strategy of observing a reference movement policy from an expert and transfer it to the real robot with the maximum possible fidelity using DRL. In order to investigate this complex scenario, this work proposes an imitation process with three phases: i) the poses estimation of a human expert based on a video of this human performing a particular tasks; ii) the generation of reference motion trajectories to a robot; iii) the robot's training in a simulated environment based on DRL technique to adapt and improve the reference movements to the new body scheme and dynamics of the robot. The investigation conducted with the Marta robot in a complex simulated environment showed that the imitation-based technique is able to make the robot kick a ball an average distance of 1m from YouTube videos.

Keywords: Imitation Learning. Humanoid Robots. Deep Reinforcement Learning. Human posture estimation.

Resumo

O estudo de robôs humanoides no campo da robótica cresceu nas últimas décadas na direção do desenvolvimento de robôs capazes de dar suporte aos humanos em muitas aplicações. A evolução das técnicas de aprendizado de máquina, particularmente a abordagem Aprendizagem por Reforço (RL), ampliou os domínios da robótica para muitas novas aplicações, com base na estratégia de reforçar o agente mediante suas interações com o ambiente. A Aprendizagem por Reforço Profundo (DRL) veio para melhorar a técnica RL permitindo a aplicação da robótica em cenários de tarefas altamente complexas. No entanto, esta abordagem é bem conhecida por duas grandes desvantagens: i) seu alto custo computacional; ii) a dificuldade em treinar o robô de forma a atingir políticas específicas que são usualmente muito difíceis de modelar. Recentemente, abordagens RL baseadas na imitação de movimentos de referência surgiram no cenário robótico. O processo de aprendizagem nesta abordagem é baseado na estratégia de observar uma política de movimentos de referência de um especialista e transferi-la para o robô real com a máxima fidelidade possível usando DRL. Para investigar este cenário complexo, este trabalho apresenta um processo de imitação em três fases: i) a estimativa da postura dos especialistas humanos com base em uma coleção de vídeos destes humanos executando tarefas particulares; ii) a geração de trajetórias de movimentos de referência para um robô; iii) o treinamento do robô baseado em técnicas de DRL capazes de adaptar os movimentos de referência para o esquema e dinâmica corporal do robô. A investigação realizada com a robô Marta em um ambiente de simulação complexo mostrou que a técnica baseada em imitação é capaz de fazê-la chutar uma bola a uma distância média de 1m a patir de vídeos disponíveis no YouTube.

Palavras-chave: Aprendizado por Imitação. Robôs humanoides. Aprendizado por Reforço Profundo. Estimativa de postura humana.

List of Figures

List of Tables

List of Abbreviations and Acronyms

API	Application Programming Interface
AI	Artificial Intelligence
AMP	Adversarial Motion Priors
ANN	Artificial Neural Networks
BC	Behavioral Cloning
CNN	Convolutional Neural Network
CoM	Center of Mass
DASS	Deterministic Action Stochastic State
DE	Differential Evolution
DoF	Degrees of Freedom
DL	Deep learning
DNN	Deep Neural Network
DRL	Deep Reinforcement Learning
FPS	Frames per Second
GA	Genetic Algorithm
GAIL	Generative Adversarial Imitation Learning
GAN	Generative Adversarial Networks
GAE	Generalized Advantage Estimator
GASI	*Grupo de Automação e Sistemas Integrados*
HMR	Human Mesh Recovery
HMMR	Human Mesh Motion Recovery
HPE	Human Pose Estimation
IL	Imitation Learning
IfO	Imitation Learning from Observation
IoU	Intersection-over-Union
IRL	Inverse Reinforcement Learning
IK	Inverse Kinematics

JC SPPE	joint-candidate Single Person Pose Estimation
KL	Kullback-Leibler
LaRoCS	*Laboratório de Robótica e Sistemas Cognitivos*
NN	Neural Network
MDP	Markov Decision Process
ML	Machine Learning
MLP	Multilayer Perception
MoCap	Motion Capture
MPJPE	Mean Per Joint Position Error
ODE	Open Dynamics Engine
PAFs	Part Affinity Fields
PCP	Percentage of Correct Parts
PCK	Percentage of Correct Keypoints
PPO	Proximal Policy Optimization
PD	Proportional Derivative
PID	Proportional Integral Derivative
RL	Reinforcement Learning
ROS	Robot Operating Systems
SAC	Soft Actor-Critic Method
Semi-SL	Semi-supervised learning
SL	Supervised Learning
SMPL	Skinned Multi-Person Linear Model
TFS	Truncated Fourier Series
TRPO	Trust-Region Policy Optimization
3D	Three-dimensions
2D	Two-Dimensions
UL	Unsupervised Learning
UNESP	*Universidade Estuadual Paulista*
UNICAMP	*Universidade Estadual de Campinas*

V-REP	Virtual Robotic Experimentation Platform
ZMP	Zero Moment Point

List of Symbols

α	learning rate parameter
β	vector of shape parameter of the 3D mesh of the SMPL body model
ε	probability of taking a random action in an ε-greedy policy
γ	discount rate parameter
$\hat{A}_t$	estimated advantage at time t
$\hat{T}$	camera translation vector in the world coordinates
$\mathscr{D}$	set of demonstration data
$\mathscr{H}(P)$	entropy measure from the the probability distribution P
$\mathscr{M}$	model function
$r_t(\theta)$	ratio of the probability under the new policy π_θ and old policy $\pi_{\theta old}$
$\nabla J(\theta_t)$	stochastic gradient - column vector of partial derivatives of $J(\theta)$ with respect to θ
Φ	vector of the 3D mesh parameters
ϕ	vector of rotation angles
π	policy (decision-making rule)
π^*	policy imitation
π_θ	policy corresponding to parameter θ
θ, θ_t	parameter vector of target policy
$\widehat{c_x}$	camera center shift parameters in x-axis
$\widehat{c_y}$	camera center shift parameters in y-axis
$\widehat{p_t}$	reference position features
$\widehat{q_t}$	reference pose features
$\widehat{v_t}$	reference pose velocity features
E_π	expectation of a random variable value under a policy π
$J(\pi)$	maximum entropy objective function
$J(\theta)$	performance measure for the policy π_θ
J_{wn}	weight factor for each of the N joints

$L^{CLIP}(\theta)$ clipped surrogate objective function

P_c point in camera frame coordinates

P_o point in world coordinates frame

p_t learner-induced position features

$Q^{\pi}(s,a)$ value of taking action a in state s under policy π

$Q^{*}(s,a)$ value of taking action a in state s under the optimal policy π

q_t learner-induced pose features

r, r_t a reward

s, s' states

s_t state at time t

s_f Scale factor

T translation vector

t_x translation parameter on the x-axis

t_y translation parameter on the y-axis

t_z translation parameter on the z-axis

V array estimates of state-value function V^{π} or V^{*}

V^{π} state-value function under policy π

$V^{\pi}(s)$ value of state s under policy π

V^{*} state-value function under optimal policy π

$V^{*}(s)$ value of state s under optimal policy π

v_t learner-induced pose velocity features

w weight vector

W_i synaptic weight vector

X_i input values in an artificial neuron

f focal length

R rotation matrix

a, a_t an action

b bias

Contents

1 INTRODUCTION

"The greatest obstacle to discovery is not ignorance - it is the illusion of knowledge."
Daniel J. Boorstin

The dream of creating machines to support human beings has long been present in humanity. This dream is gradually becoming part of our reality due to the improvement of machine skills through Artificial Intelligence (AI).

Robotics can be defined as the science that studies the intelligent connection between perception and action (SICILIANO; KHATIB, 2016). The field of humanoid robotics is concerned with the creation of robots that are directly inspired by human capabilities and aspects of human behavior. Today robots are used to support humans in many applications and are designed to cohabit with humans in homes, workplaces, schools, hospitals, and other places. Some researchers characterize the era we live in as an opportunity to understand ourselves better, our behaviors, and emotions, emphasizing areas of human knowledge that are rapidly developing, such as computer, science, and biomechanics (GOSWAMI; VADAKKEPAT, 2019).

In the humanoid robotics field, the study of performing movements while keeping a robot's balance is one of the most challenging. The first thing we need to control for bipedal robots is the balance to guarantee the robot's posture in different conditions. The main approaches currently adopted to perform this task are the Zero Moment Point (ZMP) (LIM; OH; KIM, 2012) and the inverted pendulum model (LIU; NING; CHEN, 2018).

In a different line, model-free Deep Reinforcement Learning (DRL) algorithms have been developed and applied in a range of applications, particularly in control tasks (MNIH et al., 2015). According to DONG; DING; ZHANG (2020), DRL research and applications deal with some complex challenges as : *i)* the sample efficiency problem; *ii)* stability of training; *iii)* the exploration problems; *iV)* sim-to-real transfer and the gaps between simulated environments and the real world.

The sample efficiency problem concerns the expensive training process, with massive interactions required to collect thousands of data to solve tasks that humans could solve in a shorter period. In a real-world application, its means enormous cost and challenge. Since humans learn by imitating others, this behavior inspired the creation of the Imitation Learning (IL) approach as an alternative method to solve problems with low learning efficiency in various DRL applications. The learning process in this approach is based on observing a reference policy from an expert and searching to learn a generalized policy that incorporates unexpected cases.

Imitation Learning has become a popular approach because it facilitates the teaching of complex tasks to the agent with minimal expertise, which opens a range of AI applications such as humanoid robots, human-computer interaction, self-driving vehicles that requires real-time perception, and reactions (HUSSEIN et al., 2017). The combination of RL and IL methods is applied to a range of applications, such as transferring the policies learned to the real robot with the maximum possible fidelity (SERMANET et al., 2018; XIE et al., 2019a; PENG et al., 2020a).

In practice, many issues contribute to making the transfer policy from humans to robots a challenging task: *i)* an extensive catalog of human reference movements is not available; *ii)* the procedures to transfer policies from the reference movements to the robot are not consolidated in the literature; *iii)* the body schemes of humans and robots are not identical (they differ in number and joint position of joints, size of links, weights, power, etc.); *iv)* the robot may be unbalanced, or its movements can fail even if flawlessly executing the human policy due to differences in body schemes.

In this complex scenario, this work proposes investigating new approaches to the imitation learning process, looking to improve the quality of humanoid robot movements in particular tasks. The investigation will take place with the humanoid robot Marta, developed by the working group: *Grupo de Automação e Sistemas Integráveis* (GASI) at the

Institute of Science and Technology, *Universidade Estadual Paulista* (Unesp), Campus Sorocaba, and the *Laboratório de Robótica e Sistemas Cognitivos* (LaRoCS) at *Universiade Estadual de Campinas* (Unicamp). The robot will be evaluated in performing basic actions such as kicking a ball.

1.1 Objective

1.1.1 Main Objective

The main objective of this work is to investigate new approaches and propose frameworks that may enable humanoid robots to mimic the movements of human experts performing particular tasks using DRL techniques.

1.1.2 Specific Objectives

The specific objectives of this work are:

1. **Create a movement policy dataset:** policies of human experts performing a particular task will be properly collected and organized in datasets based on the estimated pose of human experts extracted from videos of these humans;
2. **Development of an imitation learning framework:** a robot will be trained using RL techniques in a simulated environment to imitate, as faithfully as possible, the policies of the human experts, dealing with some eventual discrepancies in the body scheme of humans and robots. Particularly, the Soft Actor-Critic (SAC) algorithm (HAARNOJA et al., 2018) will be investigated in this context;

1.2 Contributions

As main contributions of the present work, we highlight:

- The development of a new dataset (*SoccerKicks*) that makes available a collection of dead-ball kick videos to provide reference movements for humanoid robots;
- The development of methodologies for choosing videos, estimating human pose, and obtaining reference movements in this dataset;
- The development of a novel methodology for training realistic humanoid robots using dataset videos based on Deep Reinforcement Learning (DRL);
- The development of an approach capable of dealing with the fact that humanoid robots can have a body shape significantly different from the humans in the reference videos;
- The improvement of the DRL algorithm, adapting it to use position coordinates instead of orientations;
- The development of new reward functions for the DRL;
- The development of frameworks that integrate distinct environments and tools like CoppeliaSim simulator, standard robotics toolkit, PyRep, RL toolkit, OpenAI Gym, and Rlkit;
- The proposition and comparison of distinct reward functions and the investigation of techniques that can stimulate more natural and realistic postures from learning.

1.3 Text Organization

This text is organized as follows:

- Chapter 1 presents the introduction and objectives of this work;
- Chapter 2 presents the theoretical background with concepts about RL, DL DRL, IL techniques and HPE;
- Chapter 3 presents related works and recent developments in the field used as a basis for the development of this project. We focused primarily on PENG et al. (2018);
- Chapter 4 details the proposed approach for transferring human movements from videos to robots;
- Chapter 5 presents the materials, methods and experimental procedures for evaluating the approach;
- Chapter 6 presents the results and discussions;
- Chapter 7 presents the conclusions and future works.

2 THEORETICAL BACKGROUND

Robots move and act in a three-dimensional and dynamic world as human beings, perceiving the world through sensors. Vision is a powerful tool and a prominent way to perceive the environment for many biological systems and robot among these sensors.

One particular class of robots very relevant nowadays is the humanoid robot. Although it has different shapes and sizes, it usually has human-like features like a head, two arms, and a biped. The robot body mechanism consists of many links (rigid parts) connected by different types of joints, with several Degrees of Freedom and sensors embodied. Each attached link by joints has an orientation and position in the world, represented by a homogeneous transformation matrix, concerning its parent joint, which describes the kinematic rigid body (JAZAR, 2010). For this reason, the stable motion control of the body represents a hugely complex engineering task.

Humanoid locomotion has been one of the most popular research areas in robotics for decades. It consists of developing controllers for a stable movement, such as balance control and walking patterns. As a control solution, in addition to traditional control methods, the advances in models inspired by human intelligence initiated a fertile era of research in Artificial Intelligence in the mid-20th century (GOSWAMI; VADAKKEPAT, 2019).

Machine Learning is a sub-area of Artificial Intelligence, splited into four mains types of learning (RUSSEL; P, 2013): Unsupervised Learning (UL), Supervised Learning (SL), Semi-supervised learning (Semi-SL), and Reinforcement Learning (RL).

In UL, the agent learns through interactions, deriving patterns from the input without explicit annotation. With RL, the agents learn from reinforcements as rewards or punishments through interactions in the environment. In SL, the choice of actions the agent will take is supervised by examples previously labeled by an expert – input-output pairs. Hence, learning is a function mapping from it. Semi-SL works between SL and UL. In SL, the agent receives a few labeled examples and deals with a considerable collection of unlabeled examples.

With the advances in ML, computer vision has expanded. It now supports image classification, object detection, Human Pose Estimation, Neural style transfer, and face recognition. Computer vision in robotics has extended the possibilities of applications and development of mobile robots such as self-driving cars, underwater vehicles, surgical robots, and manufacturing robots (CORKE, 2017).

Suppose a robot is equipped with a camera. In that case, we could adopt different models and methods based on ML and mathematical modeling to extract different pieces of information from the world. In particular, a robot could watch a human performing a task while mimicking its behavior.

Given this brief introduction, his chapter will detail the theoretical background required to create a framework that can imitate Human Behavior from video frames. In the context of ML, we will present the model-free control by RL (section 2.1) and its description in terms of Markov Decision Processes (section 2.2). We will also address Deep Learning (section 2.3) as well as state-of-the-art DRL algorithms (section 2.3.1). Finally, we will introduce imitation learning (section 2.4) and the human pose estimation approach (section 2.5).

2.1 Reinforcement Learning

Nowadays, Reinforcement Learning (RL) is one of the most popular paradigms for developing agents that can accomplish challenging tasks in uncertain environments. RL is a sub-area of ML concerned with how agents will interact with the environment and act to learn a behavior that can maximize a cumulative reward. The agent chooses an action based on the policies learned during the training process. The agent can optimize accumulated reward values to an

optimal policy. The Figure 1 is a diagram of the RL process. The RL process consists of an agent that is the learner in an environment, and its interactions are observed by a state and measured by a scalar reward signal.

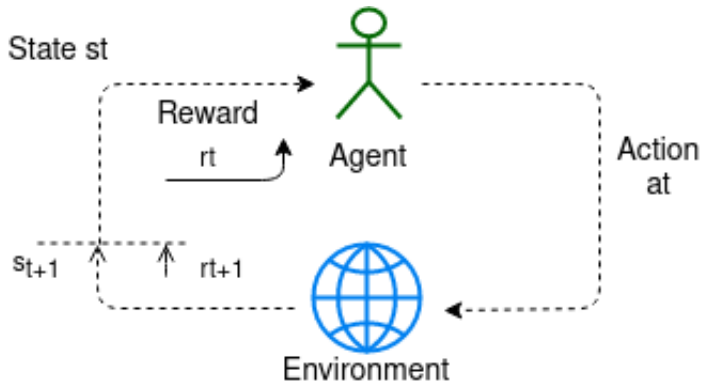


Figure 1 – The diagram of reinforcement learning process. In this diagram the Agent at time t takes an action (a_t) according to the current state (s_t), passing to a transitory state, and receives a reward r_t given the current state as a feedback to the agent about its action. The cycle continues until a certain condition is reached.

The RL architecture is formed by:

- *Agent*: is the learner;
- *Environment*: is the environment in which the agent interacts, given by:

  *State (**s**)*: defines the current state of the environment;

  State transition (s'): defines the current state after the agent acts and the relative reward which the agent will receive;

  *Action (**a**)*: defines the available actions for the agent in a given state;

  *Reward signal (**r**)*: defines the numerical reward the agent receives after interacting with the environment. The reward signal indicates the good or bad events for the agent. It depends on the agent's current action and the current state of the agent's environment.

- *Policy* (π): defines the way of behaving, at a given time. Then, $\pi(s,a)$ is the probability which the agent will choose an action a given the state s following a policy π;
- *Value function (V)*: defines the quality of states in the long-term, taking into account the possible rewards in future states. While the reward only gives the immediate reward to the agent, the value function, V_π in a state s is the long-term quality of state s when following policy π;
- *Model*: A model predicts the next behavior of the environment. The model is used to plan in the environment. It predicts the next reward or the resulting state, given a state and an action.

RL methods that solve the problem by employing the model are called model-based methods. Model-based RL aims to train a model given the state-action pair (s_t, a_t) to predict the next state s_{t+1}. Once trained, the agent can use the planning directly through the model to choose its actions, maximizing the expected rewards. Dynamic programming aims to find the optimal policy given a model of the environment. In contrast, model-free methods approach learning by trial and error without knowing the model dynamics. Given a state-action (s_t, a_t) pair, the agent has to assess the

consequences of its action – the next state s_{t+1} and the reward – by directly trying it in the environment. In both contexts, RL methods are inspired by studies in human behavior in which the rewards are comparable with pleasant or painful experiences, given by the action in the environment and their current state.

Therefore, the agent must be sensitive to the state in the environment and perform actions that affect this state to achieve the goal defined in the environment (SUTTON; BARTO, 2018). A RL problem can be formulated mathematically by Markov Decision Processes (MDP), described next.

2.2 Markov Decision Process

Formally, the RL problem can be described as a Markov Decision Process (MDP) satisfying the Markov property. Considering an observable environment, at time t given any state s and action a, the probability distribution of each possible pair of next state s' and reward r at time $t+1$ is denoted by:

$$Pr\{R_{t+1} = r, S_{t+1} = s' | S_0, A_0, R_1, ..., S_{t-1}, A_{t-1}, R_t, S_t, A_t\} \tag{1}$$

$$= p(s', r|s, a) = Pr\{R_{t+1} = r, S_{t+1} = s' | S_t = s, A_t = a\}. \tag{2}$$

The equality between equations (1) and (2) represents an environment that has the Markov property. An environment that satisfies the Markov property allows the choice of the best action related to predicting all future states and the expected rewards solely from the observation of the current state.

Most solutions to RL problems are model-free methods, where search optimizes the policy based on trial and error into the environment. These methods are split into two main approaches: Value-Based Methods and Policy-Based Methods.

2.2.1 Value-Based RL

The parameter about how good is to perform an action (or the expected return of that action) in a given state s for the agent, is described by a value function $V^{\pi}(s)$, under an arbitrary policy π. Then, the Value-based method aims to learn the state Value function ($V^{\pi}(s)$ and $V^{*}(s)$) or the state-action Value function ($Q^{\pi}(s,a)$ and $Q^{*}(s,a)$), by the following equations:

$$V^{\pi}(s) = E_{\pi}[\sum_{i=0}^{\infty} \gamma^i R_{t+i} | S_t = s]. \tag{3}$$

$E_{\pi}[.]$ is the expected value for the randomness of future actions according to the policy, and γ is the discount factor that can favor immediate rewards over future ones. $V_{\pi}(s)$ considers the current state and average across all future actions that can be taken (MARSLAND, 2015). Similarly, when we consider the current state and each possible action individually, we can define the state-action Value function($Q^{\pi}(s,a)$ and $Q^{*}(s,a)$):

$$Q^{\pi}(s,a) = E_{\pi}[\sum_{i=0}^{\infty} \gamma^i R_{t+i} | S_t = s, A_t = a]. \tag{4}$$

From the definition of the expected return (Eq.3), we can obtain the optimal expected return as:

$$V^{*}(s) = max_{\pi} V^{\pi}(s). \tag{5}$$

Correspondingly, we can define the optimal Q-value function $Q^{*}(s,a)$ as:

$$Q^{*}(s,a) = max_{\pi} Q^{\pi}(s,a). \tag{6}$$

From the Bellman Equation, equation 4 can be rewritten recursively as an MDP by:

$$Q^{\pi}(s,a) = \sum_{s' \varepsilon S} T(s,a,s')(R(s,a,s') + \gamma Q^{\pi}(s', a = \pi(s'))). \tag{7}$$

The Bellman Equation is an important formula to describe the relationship between the immediate value of a state and the values of its successor state from the expected total reward by taking action from a state in an MDP.

The value-based methods as Monte Carlo and Temporal-Difference (TD) are part of the theoretical background to understand and solve many problems of RL (SUTTON; BARTO, 2018). Between these approaches, the TD algorithms are one of the most widely applied methods in RL problems, with algorithms as SARSA (On-policy) and Q-learning (Off-policy).

The On-policy methods attempt to evaluate or improve the same policy (following a policy) that is used to explore the environment. In contrast, the Off-policy methods evaluate or improve different policies (not following a policy) to explore the environment. In line with these different methods, this section will discuss the Q-learning algorithm.

Q-learning Algorithm (WATKINS; DAYAN, 1992) is a form of model-free off-policy RL belonging to the class of TD method (MITCHELL et al., 1997). The learned action-value function (Q-function), through experimentation in the environment, searches an optimal estimation for the action-value function through an exploration strategic, e.g., ε-greedy policy.

The Q-function $Q^{\pi}(s,a)_t$ (Eq. 8) denotes the expected accumulated reward obtained by selecting an action given by the current state, following policy π. It provides agents capable of learning to act optimally in Markovian domains by experiencing the consequences of actions without prior knowledge, defined by.

$$Q(s,a)_{t+1} \leftarrow Q(s,a)_t + \alpha[r + \gamma \max_a Q(s',a)_t - Q(s,a)_t, \tag{8}$$

where:

- $\max_a$ is the maximum return that is attainable in the state following the current one, i.e, the reward for taking the optimal action thereafter.
- α (alpha) is the learning rate ($0 < \alpha \leq 1$);
- γ (gamma) is the discount factor ($0 \leq \gamma < 1$) which determines how much importance we want to give to future rewards.
- $\max_{a'} Q(s',a)_t$ refers an exploration strategy to select the optimal action with respect to the state (s') using the policy derived from Q, such as $\varepsilon - \textbf{\textit{greedy}}$ (epsilon -greedy) which is a balanced strategy of exploration and exploitation for selecting actions (the best action or a random action) with a fixed probability $0 \leq \varepsilon \leq 1$.

A high value for the discount factor (close to 1) captures the long-term effective reward. Whereas, a discount factor of 0 makes the agent consider only immediate rewards, hence making it greedy.

2.2.2 Policy-Based RL

Policy-Based RL methods directly learn an optimal policy π parametrized by vector θ (Eq. 9) by selecting actions in a given state of an environment without consulting a value function, as:

$$\pi(a|s,\theta) = Pr\{A_t = a | S_t = s, \theta_t = \theta\}. \tag{9}$$

Policy-based methods are divided into two categories:

Policy Iteration Method: This method is composed of two interacting processes in an attempt to evaluate and improve the policy until finding an optimal policy. Given the initial policy π_0, chosen arbitrarily, the policy evolution is applied ($\xrightarrow{E}$) by using the Bellman equation (Equation 7) to estimate the current value function V^{π_0}. Then, the next best state decision based on $V(s_t)$ leads on the policy improvement ($\xrightarrow{I}$), which generates a better policy π_1 by making it greedy concerning the value function of the previous policy. The same operation is applied N times to yield the optimal policy π_* and the v_{π_*} (SUTTON; BARTO, 2018).

$$\pi_0 \xrightarrow{E} v_{\pi_0} \xrightarrow{I} \pi_1 \xrightarrow{E_N} \ldots \xrightarrow{I_N} \pi_* \xrightarrow{E} v_{\pi_*}. \tag{10}$$

Policy-Gradient method: Policy-Gradient methods employ the gradient (or an estimator) to learn a parameterized policy. Given any performance measure $J(\theta)$ parametrized by θ, the policy gradient searches for a local maximum in $J(\theta)$ by ascending the gradient of the policy in J:

$$\theta_{t+1} = \theta_t + \alpha\widehat{\nabla J(\theta_t)}. \tag{11}$$

PG methods maximize their performance (the expected total reward) by repeatedly estimating the stochastic gradient $\nabla J(\theta_t)$ with respect to the parameter θ_t at time t (SUTTON; BARTO, 2018).

2.2.3 Actor-Critic methods

Actor-Critic methods aim to combine the advantages of value-based and policy-based methods, attempting to reduce the variance of policy gradients by evaluating the current policy given the value function, and then update the policy parameters following a performance improvement direction (WU et al., 2019). The actor is the policy structure that uses it to select the actions, and the critic criticizes the actions executed by the actor with the estimated value function.

After each action selection, the critic is a state-value function and updates the weight vector **w** (evaluating the current policy) using the TD error signal. Then, the state-value function (critic) updates the *actor* policy parameters θ to optimize the cumulative reward (improving the current policy) (GRONDMAN et al., 2012), as the Figure 2 represents.

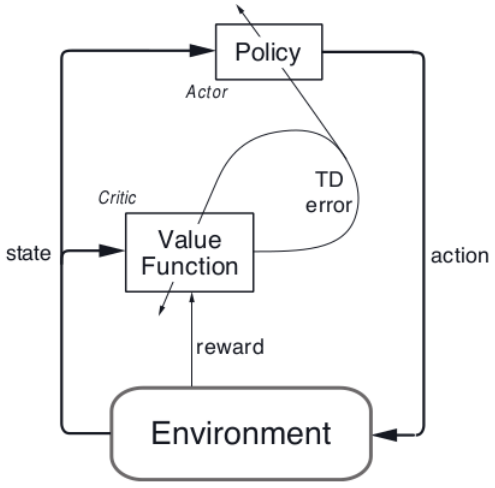


Figure 2 – Actor-Critic architecture. The actor is the policy to select the actions, which are evaluated given the TD error in the estimated value function, the critic. Then, TD error can improve the action selection. Extract from SUTTON; BARTO (2015).

2.3 Deep Learning

Figure 3 presents a biological neuron and its artificial representation in the context of the Artificial Neural Networks (ANNs). A biological neuron is typically connected to a number of other neurons in the brain. It receives signals – electrical impulses – in its dendrites collected from axon terminals of other neurons. These signals are processed in internal connections generating an output that may be used to trigger other neurons or to activate some muscle.

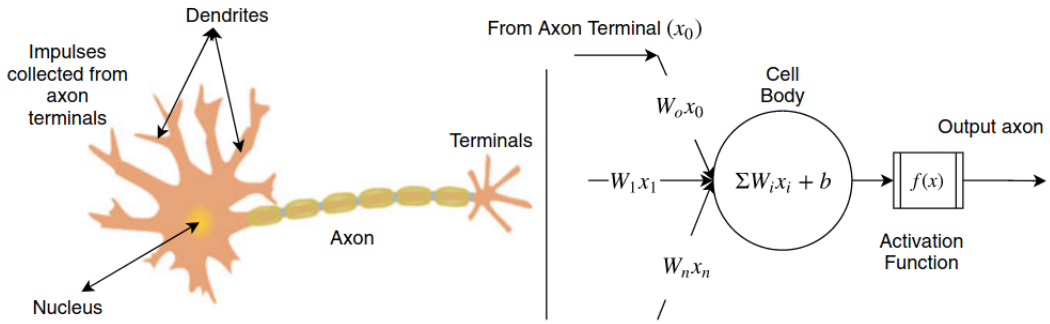


Figure 3 – Representation of a biological neuron (left) and the artificial neuron model (right). Extract from PEDRYCZ; CHEN (2020a).

In an artificial neuron the input values (X_i) is a vector in the form $X_i = \{X_0, X_1, ..., X_n\}$. The inputs are weighted using a synaptic weights vector (W_i), with $W_i = \{W_0, W_1, ..., W_n\}$, that will be tuned in the learning process. In cell body, the neuron will trigger typically if the sum of elements $\sum W_i X_i + b$ is greater than a threshold, where b is a bias, that is, a real number that shifts the value of the output. The frequency of trigger is modeled by an activation function $f(x)$. The type of Activation function explicitly affects the performance of NN. The popular Activation functions are linear, sigmoid and softmax, Tanh (Hyperbolic Tan), ReLU family. For more details (PEDRYCZ; CHEN, 2020a). Other important element of NN is the loss function, that is, a cost function applied over actual and predicted values. The loss function is measured during the training in order to minimize the error over the training.

ANNs are organized in layers. The *perceptron*, a single-layer network, and its multilayer version, the *Multilayer Perceptron* (MLP), are some of the most common ANNs. In a network, the layer is called **dense layer** or **fully connected layer** in case each neuron is connected with all neurons of other layer. The MLP has at least two dense layers of neurons, usually with more than one hidden layers between the input and the output layers. The layers can have different number of neurons according to the network application. The Deep Neural Network (DNN) is an ANN with multiple – usually a large number of – layers between the input and output layer, as show in Figure 4.

When the activation signal propagates from the neurons in one layer to those in the next layer, the network is called **Feed-forward DNN**. The synaptic weights in each layer are individually updated by an algorithm considering the gradient of the loss function, gradually reducing the network error. In other words, the **Backpropagation** allows the calculation of gradients from backward.

DNNs are usually characterized into one of three different categories: Multilayer Perception, Recurrent Neural Networks (RNNs) and **Convolutional Neural Networks** (CNNs), which are particularly relevant in this work. CNNs are inspired in the Visual Cortex of the human brain, and are widely applied to process data with multiple arrays in order to differentiate various aspects of images. Figure 5 shows a representation of a feed-forward CNN based on 3D neural arrangements, emphasizing the fully connected layer.

The typical CNN processing can be described in three steps. In the beginning, the *Input Layer* is a full image (video frame), that is, it is a tensor with the dimension of the image (width x height x color-channels) that can have an

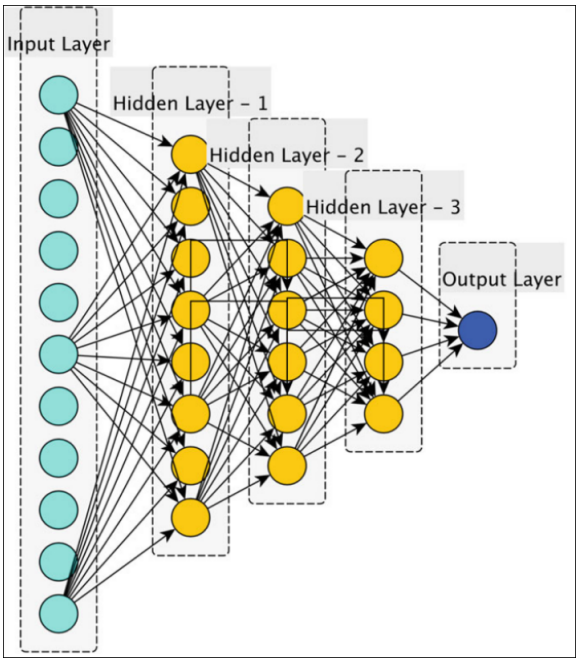


Figure 4 – Representation of DNN with three hidden layer and one node in output layer. Extract from SEWAK (2019).

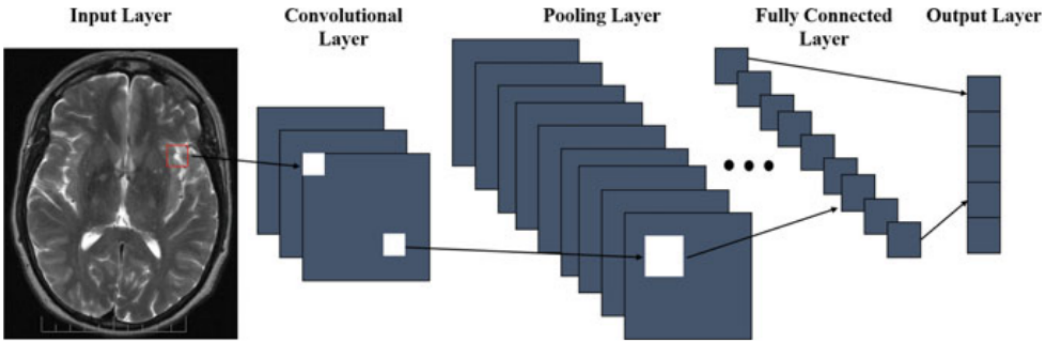


Figure 5 – A condensed representation of a feed-forward CNN. The Figure presents the common layers for a CNN: Input layer, Convolutional layer, Pooling layer and the Output layer. All the layers are fully connected with each other. Extract from PEDRYCZ; CHEN (2020b).

additional dimension as a sample index or batch.

In each *Convolutional Layer* (also called a detector stage), multiple convolutions are generated based on the application of a kernel (or filter mask) over the image. The number of kernel filters that can differ from layer to layer. As the output, each of them generates a set of objects or feature maps. The feature map generated by a single kernel filter can be used to define a new input to the next layer. In convolutional layers, typically each neuron contains all the information of a small region responsible for different tasks such as edge, color, or angle detection, through all channels.

The *Pooling Layer* acts across the dimension of subregions of each convolutional map generated from the previous convolutional layer. There are two main types of polling layers: max-pooling and average-pooling. The pooling

technique reduces the output size and, therefore, reduces the computational complexity of the next convolutional layers. As a result, the height and width of each convolutional map is reduced after the pooling layer.

All the previous layers are typically fully connected to each other, as the hidden layers of a DNN. Until the network is able to achieve a result, different weights are considered (PEDRYCZ; CHEN, 2020b).

According to PEDRYCZ; CHEN (2020b), CNN has two main properties: sparse interactions and parameter sharing. For example, in a high-resolution image classification application, the kernel will be configured to capture only key features like contrast and edges in the image. These properties of sparse interactions can reduce the parameters to represent the image, resulting in a reduction of memory and computational process. Also, in the image classification task, the parameter sharing by tied weights guarantees that only one set of parameters is learned for each location of the image.

Deep learning, as a machine learning tool, has achieved enormous success in applications in computer vision, natural language processing, robotics, and so on. In this scenario, the next section will address some algorithms adopted in Deep Learning with Reinforcement Learning, also called as Deep Reinforcement Learning.

2.3.1 Deep Reinforcement Learning

Advances in RL aided by DNNs have solved very complex tasks. Such things were possible because DRL has high dimensional observations in state-space, allowing the agent to make better decisions (ARULKUMARAN et al., 2017).

Popular algorithms of Deep learning such as Trust-Region Policy Optimization (TRPO), Actor-Critic (AC), Synchronous Advantage Actor-critic (A2C), Asynchronous Advantage Actor-critic (A3C), and Deep Deterministic Policy Gradient (DDPG), are policy gradient methods that use Neural Networks as function approximators.

In this context, this section introduces the state-of-the-art DRL algorithms: on-policy Proximal Policy Optimization (PPO) and off-policy Soft Actor-Critic (SAC).

Proximal Policy Optimization

Proximal Policy Optimization (PPO) introduced by SCHULMAN et al. (2017) is an on-policy method. It is a variant of TRPO where the algorithm's stability and credibility are maintained while the model is simplified. The PPO's main functions are the adaptative Kullback-Leibler (KL) penalty and the clipped objective. First, Eq. 12 introduces the probability ratio between the current policy (π_θ) and the policy before the update ($\pi_{\theta old}$):

$$r_t(\theta) = \frac{\pi_\theta(a_t, s_t)}{\pi_{\theta old}(a_t, s_t)}. \tag{12}$$

PPO updates each policy gradient to minimize the cost function at each step, ensuring a small deviation from the previous policy. The KL quantifies the divergence between the probability distribution and is used to control the policy updates at each iteration. The clipped objective function is given by Eq. 13. It prevents that the new policy and the old one diverge too much.

$$L^{CLIP}(\theta) = \hat{E}_t[min(r(\theta)\hat{A}_t, clip(r(\theta), 1-\varepsilon, 1+\varepsilon))\hat{A}_t]. \tag{13}$$

Given the policy parameters θ, $\hat{E}_t$ is the expectation at time t. $\hat{A}_t$ is the advantage function (the expected rewards minus a baseline like $V(s)$) for the new policy, and ε is a hyperparameter. The probability r_t is clipped at $1-\varepsilon$ or $1+\varepsilon$ for positive or negative values, respectively.

Soft Actor-Critic Algorithm

Soft Actor-Critic Method (SAC) is an off-policy actor-critic DRL method proposed by HAARNOJA et al. (2018), where the actor's goal is to maximize the expected reward and entropy. With the adoption of two soft Q-functions, SAC mitigates the positive bias in the policy improvement step and trains them independently for parameter optimization. Then, the minimum of the Q-functions is used in the value and policy gradient. SAC is an off-policy algorithm that optimizes a stochastic policy.

The entropy $\mathscr{H}$ can be defined as how unpredictable a random variable is, and is measured from its distribution P by:

$$\mathscr{H}(P) = E_{x\sim P}[-\log P(x)]. \tag{14}$$

The entropy is regulated with the coefficient non-negative temperature α, exercising stochastic control of the optimal policy that direct affects the reward. High α means more exploration, and lower α corresponds to more exploitation. The function that aims to maximize its entropy at each visited state is define by:

$$J(\pi) = \arg\max_{\pi} \sum_{t=0}^{T} E_{(s_t,a_t)\sim p_\pi}[r_{(s_t,a_t)} + \alpha\mathscr{H}(\pi(\cdot|s_t))]. \tag{15}$$

Similarly, the equations of V^ϕ and and Q^ϕ also include the entropy bonuses.

2.4 Imitation Learning

Traditional methods for programming autonomous behavior in robots normally require specific knowledge and a huge amount of time for training, with results often inferior to those expected, especially in the reproduction of human movements. Therefore, the Imitation Learning method (IL) has the purpose of efficiently learning a required skill by imitating the expert's ability. Usually, IL is a technique that enables skills to be transferred from human operators (the experts) to robotic systems (the learners) (OSA et al., 2018).

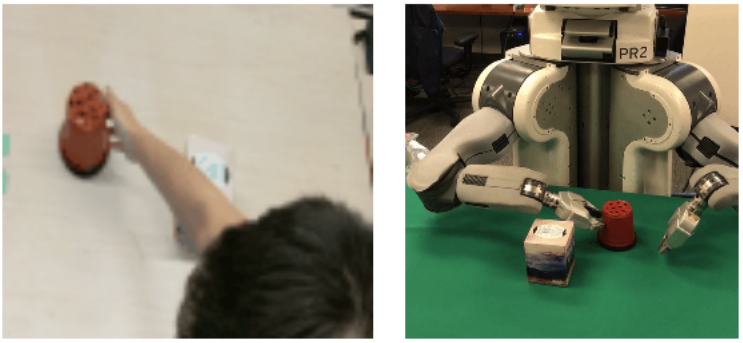


Figure 6 – Transfer of the ability to push the vessel from the expert human to the robot. Extract from YU; FINN (2018).

IL consists in a set of expert demonstrations to guide the agent to learn an optimal policy imitating the expert's. The simplest way to apply IL is through a collection of sample trajectory demonstrations, where τ is a sequence of features $\tau = \Phi_0$, their context information $\mathbf{S}$, and an optional reward signal $\mathbf{r}$. All these elements are incorporated in a dataset $\mathscr{D} = (\tau_i, S_i, r_i)_{i=1}^N$. The optimization-based strategy learns a policy π^* that relates the distribution of the characteristics induced by the expert policy ($\widehat{p_\pi}$) and the distribution of the characteristics induced by the learner (p_π), that minimizes training error. Given the similarity measure $\mathscr{D}(\widehat{p_\pi}, p_\pi)$, that satisfies:

$$\pi^* = \arg\min \mathscr{D}(\widehat{p_\pi}, p_\pi). \tag{16}$$

Since we learn a policy π from a dataset $\mathscr{D}$ (demonstrations), this setting is closely related to the supervised learning method. However, the IL methods differ from the well-known supervised learning environment since the agent tries to learn from experiences, normally in an RL environment. This means that even if the action performed is not optimal, the agent receives a reward by performing that action in a certain state. In other words, the agent attempts to extract the underlying principles for generating good actions from the demonstration dataset and apply them to more general cases (DONG; DING; ZHANG, 2020). Even if the demonstration dataset has good or bad examples, it reduces the complexity of search spaces for learning.

Recent advances have shown that IL methods may be essential for robotics (OMPICO; BUGTAI; MUNSAYAC, 2021; ZAHRA et al., 2022). Imitation Learning methods can be divided into two categories depending on the information contained in the expert reference. Therefore, the categories are IL from demonstration and IL from observation. The main difference between the categories is that learning from demonstration requires both states and actions, while learning from observation demands only the states' information.

2.4.1 Imitation Learning from Demonstration

IL from demonstrations utilizes both state and actions to learn and is broadly divided into Behavioral Cloning and Inverse Reinforcement Learning.

Behavioral Cloning (BC): is the simplest form of IL, which aims to clone an expert's behavior given expert trajectories as a supervised learning task, considering the states as the inputs and actions as the labels.

However, it deals with some challenges. The data distribution shifts in the new application as the demonstration dataset contains finite samples. Hence, learning can fall into the decision-making problem of mismatched state distribution between training and testing. Also, the BC can accumulate errors over time which can induce a profoundly different state distribution and lead to a critical failure. To solve this issue, LASKEY et al. (2017) inject noise into demonstrated policies to learn how to recover from errors. Another method proposed by BRANTLEY; SUN; HENAFF (2019) uses the variance of the predictions from a set of policies trained on the expert sample data as a cost, minimizing it with RL.

Inverse Reinforcement Learning (IRL): infers a hidden reward from the optimal behavior (expert demonstrations) and then uses RL to find the optimal policy that maximizes the reward function learned – the imitation policy. IRL method encourages the agent to solve an MDP to find the optimal policy.

The steps to achieve this goal are repeated several times, which can be deeply expensive from a computational perspective. To mitigate this problem, HO; ERMON (2016) propose the framework Generative Adversarial Imitation Learning (GAIL) based in Generative Adversarial Networks (GAN) (GOODFELLOW et al., 2014) architecture. Another challenge is the ambiguity of the reward, where the multiple reward/cost functions are relative to the same optimal behavior demonstrated. The reward ambiguity is solved using maximum entropy, as demonstrated by ZIEBART et al. (2008).

2.4.2 Imitation Learning from Observation

Since humans learn skills through observation, learning by observing states seems a more natural way to reproduce human behavior. Then, Imitation Learning from Observation (IfO) is an orthogonal approach that focuses on the problem of unobservable actions, which the agent learning only from state demonstrations, i.e., $\tau = \{o_t\}$ generated by an expert to imitate their actions.

This method came to accomplish tasks where it is infeasible to obtain the actions from the expert in various practical scenarios, such as cloning humans walking by robot due to the absence of human actions. In contrast, the robot could learn by exploiting the information from videos of a human expert with IfO, just as this dissertation aims to do. In

another scenario, directly imitating actions in different environment dynamics between the learner and the expert would lead, e.g., to a control failure (CHENG et al., 2021).

Following TORABI; WARNELL; STONE (2019), the IfO approach has two main embedded components, which are: perception and control.

Perception

Perception is about how the observation states will be perceived. Recent approaches, through advances in visual recognition, use raw video information as an expert demonstration (LIU et al., 2018). Two main problems are recurrent with the perception, which are: embodiment mismatch and viewpoint difference.

1. **Embodiment Mismatch:** is about the differences in embodiment features (e.g., appearances, dynamics) between the expert to the imitator. For example, the difference between the robot dynamics that attempt to imitate human dynamics to perform a task from videos. To address this problem, GUPTA et al. (2017) use autoencoders to learn a correspondence between embodiments in a supervised model.

2. **Viewpoint Difference:** is about the viewpoint from the camera in an image or video not recorded in a controlled environment.

Control

The control is the approach used to learn the imitation policy as an IfO problem. In the literature, we can organize the IfO algorithms into two wide groups: Model-based and Model-Free algorithms.

1. **Model-Based.** The dynamics of the environment can be learned with a model-based approach during the imitation process. These methods can be categorized as follows:

   - Inverse Dynamics Models: is a mapping from state-transitions (s_t, s_{t+1}) to actions a_t. In an exploratory manner, the algorithm collects the data through the environment to be used to learn the mapping of observed transitions to the actions for the imitator (NAIR et al., 2017)
   - Forward dynamics Model: is a mapping from state-action pairs (s_t, a_t) to state-transition s_{t+1}. EDWARDS et al.(2019), propose an algorithm called Imitating Latent Policies from Observation to address this method.

2. **Model-Free.** These algorithms attempt to learn the imitation policy without a model of the environment. These methods can be:

   - Generative Adversarial Methods: is based on the GAIL. The generator is the imitation policy, and the data collected from the policy execution in the environment feed the discriminator. The discriminator differentiates between the data that comes from the imitator and data that comes from the reference. Then, the imitation policy is updated under RL settings taking into account the output signals from the discriminator (MEREL et al., 2017).
   - Reward-engineering Methods is modeling the reward function based on expert demonstrations to find the imitation policies in the RL setting. A common approach for this method is to find the Euclidean distance between the reference and the agent's state at each time step, as shown (KIMURA et al., 2018; SERMANET et al., 2018).

2.5 Human Pose Estimation (HPE)

The Human Pose Estimation (HPE) aims to locate the joints that form the posture of the human body from input images or video sequences. HPE systems provide motion and geometric information of the human dynamics that can be applied in different tasks in computer vision such as Human-computer interaction, Augmented Reality, Virtual Reality, and Human tracking.

The human articulation is also known as keypoint. It is located by a set of coordinates that, connected, describe a person's pose. For each body part in an RGB image, the pose can be estimated in Two-Dimensions or Three-dimensions in space. The number of keypoints can vary between systems, estimating the foot, hand, and face points. To illustrate the problem, Fig. 7 presents the humans in poses with 14 highlighted keypoints that are interconnected, forming the pose tree.

Figure 7 – The 14 keypoints are the white points in each joints as present in each human pose tree. Extract from BEARMAN; DONG (2015).

In recent years, Deep Learning techniques have brought significant progress in the HPE field, generally applied with CNN architectures. Classical approaches to HPE applied the called pictorial structures by tree-structured graphical models (ANDRILUKA; ROTH; SCHIELE, 2009; PISHCHULIN et al., 2013). Other approaches applied Hand-crafted features (YANG; RAMANAN, 2011; WANG; LI, 2013). The HPE systems use distinct strategies to perform the pose estimation by adopting different methods, and models (CHEN; TIAN; HE, 2020).

1. The methods can be defined as:

   - **Generative:** the human body is based on a model. **Discriminative:** the methods are model-free.
   - **Top-down:** this method first detects people at the input. It then generates the location of the people given the constructed bounding boxes. **Bottom-up:** first, all the body parts of all the persons at the input are predicted. Then, the grouping of the parts is formed for each possible person detected.
   - **Regression-based:** direct map the input inferring the coordinates of body joints of the human body. **Detection-based:** uses representations such as image patches and heatmaps of the joint location to target the body parts.
   - **One-stage:** is an end-to-end system to predict the human pose without intermediary prediction. **Multi-stage:** is made of multi-stage methods employing intermediate supervision.

2. The Human body models are commonly fitted in three types:

- **Skeleton-based Model:** is a kinematic model that represents a set of joint locations and their orientations by a skeletal structure of the human body.
- **Contour-based Model:** provides information of the limbs and torso through raw contours.
- **Volume-Based Model:** applied in 3D HPE that represents a human body by geometric shapes or meshes.

Several **Evaluation metrics** are applied in HPE systems. Percentage of Correct Parts, for instance, measures the detection rate of limbs; Percentage of Correct Keypoints uses a threshold to measure the accuracy of localization of different keypoints; In 3D HPE - Mean Per Joint Position Error calculates the Euclidean Distance between the ground truth position and the estimated 3D joints.

Despite many years of research, the HPE still represents a complex problem. Beyond the structure of the systems as insufficient training data and depth ambiguities in 3D HPE, the high dimensionality aspects of the input images or videos can deeply impact HPE. These aspects can be the significant variations in camera movement, changes in position, object appearance, scale, point of view, rotation, illumination conditions, body occlusion. Particular challenges in this scenario involve predicting the keypoints of small joints, dealing with limb occlusion, and the need to capture the entire context of video to estimate the pose (KUEHNE et al., 2011; MAHMOOD et al., 2019).

2.5.1 Datasets

Recently, efforts are being made to catalog a variety of human motions in a diversity of databases for many purposes, especially for HPE tasks, the joint annotations of human body location are needed during the train, test, and evaluate the HPE systems. For the 2D HPE models the Microsoft-Common Objects in Context (MS-COCO or COCO) (LIN et al., 2014) is an extensive dataset suitable for a range of applications, with a large variety of human poses, with different body scales, occlusion patterns, and a large set with human body annotations for 17 joints. The MPII Human Pose dataset (ANDRILUKA et al., 2014) is also commonly used in HPE with 25K images in 410 different human activities, and the poses are manually annotated with up to 16 body joints. Finally, the Leeds Sports Pose dataset (LSP) (JOHNSON; EVERINGHAM, 2010) mostly has images of sports activities collected from Flickr searches in which each person is labeled with 14 body joints.

For the 3D HPE, the acquisition of accurate 3D annotation requires sophisticated techniques such as Motion Capture. The MoCap systems collect the motions in controlled lab environments to supply the pose annotations needed in HPE systems. Despite their quality, they are costly and have limitations for wild environments. The MoCaps datasets: Human3.6 (IONESCU et al., 2013) present 3.6 million 3D human poses and 24 joints annotations; and the 3D Poses in the Wild (3DPW) (MARCARD et al., 2018) have 60 video sequences and 15 joints annotations.

3 RELATED WORK

Deep Reinforcement Learning has been widely adopted to generate Artificial Intelligence systems with greater data representation capabilities, crossing barriers in the AI era, as achieved in the development of AlphaGo (SILVER et al., 2016). Since then, it has been applied to solve problems in different fields such as a robot control system, autonomous driving, air vehicles, telecommunications, Internet of Things, biological data, economic data, energy system management, and others.

In the robotics field, the control of humanoid robots is a complex task, particularly considering that robots differ in number and positions of DOFs, size of links, weights, and others. In particular, the RL application on bipedal robots has been limited by the difficulty to develop suitable reward functions that can lead to an efficient control with natural movements of the robots.

Since the pioneer work of BENBRAHIM; FRANKLIN (1997) in using RL to control bipedal robots, others have attempted to achieve this goal by controlling the robot's balance during walking. KIM; LEE; SENTIS (2017) adopted the actor-critic DRL-based methods to control the whole body of a walking robot modeled as a linear inverted pendulum. The simulated humanoid robot Valkyrie has 26 DoF and it was simulated in SrLib environment.

YU et al. (2019) achieved success in transferring learning to the Darwin OP2 robot for forward, backward and sideways movement with the policies learned by the PPO algorithm. Further work (YU, 2020) studied a set of algorithms with progress in learning autonomous and generalization of the locomotion for simulated characters and real robots. Authors also applied DRL algorithms to train the locomotion controller and transfer it to the real bipedal robot Darwin OP2 with 20 DoF and to the quadrupled robot Ghost Robotics Minitaur.

TOMAZELA (2019) developed a bipedal walking algorithm applied to the robot model Marta (better described in Section 4.5.1) as well as for the Atlas robot from Boston Dynamics, with 28 hydraulic joints, both with a similar design. Both robots were simulated in the MuJoCo physics engine environment. The autor's approach combines two robotic control models based on torque. The trunk of Marta robot and its spherical pelvis joint were controlled separately from the legs with the model-free DRL SAC algorithm (Section 2.3.1) to learn the policy based on Inverted Pendulum model to maintain the robot's stability by controlling the angular joints of the trunk and its torque. Marta's lower body part was controlled with model-based inverse kinematic and Jacobin calculations, which implemented a numerical solution for the robot's leg and traced the joint position at the desired speed with the controller Proportional Derivative. However, this approach is not easily transferred to a real robot since the algorithm does not work quickly for real-time and frequency calculations. Moreover, Tomazela proposed an evolutionary algorithm Differential Evolution (DE), in order to optimize the parameters of the gait combines with the DRL method.

BEGAZO (2020) proposed a decoupled robot control strategy to Marta robot, with a classical Proportional Integral Derivative (PID) controller in robot waist focusing on controlling the robot's stability, and a trajectory controller based on Truncated Fourier Series (TFS) parameterized by a Genetic Algorithm (GA) approach for controlling the remains DOFs of the robot. The 3-DOF spherical joint at the robot waist was controlled by a PID controller using the Ziegler-Nichols method. The author adopted a RL approach with Q-learning algorithm to tune the PID controller at the Hip Pitch joint to achieve the robot stability as an Inverted Pendulum model. Author also performed five experiments in the Virtual Robotic Experimentation Platform (V-REP) observing the Learning phase. The experiments consisted in the application of a simple Proportional (P) Controller with only the pitch joint released, the PID controller alternating the actuation of the joints at the waist, and with all waist joints released. As a result, achieve robustness to the bipedal robot walking while compared to traditional techniques.

SOARES et al. (2020) develop a stable walking gait algorithm with DRL for the humanoid robot Marta, with

experiments in the CoppeliaSim and PyBullet. Soares has noted that in his experiments the use of SAC algorithm to train the Marta robot reduced the training time if compared to PPO algorithm. The experiments was developed with OpenAI Gym interface and PyRep to CoppeliaSim.

In general lines, DRL algorithms have low learning efficiency, which usually requires an immense wait time with a computational power cost for a typical training process, even wrapped with frameworks to accelerate learning. These more stringent requirements on learning efficiency of DRL methods stimulate the investigation for new approaches such as the imitation learning. Since this work has the goal of create a movement policy dataset, we reviewed few related works in HPE field that utilizes DNN.

3.1 Imitation learning approaches with DRL

As described in the 2.4 section, IL methods are intended to enable efficient learning from an expert for a certain desired behavior. Given a set of demonstrations, the agent has a reference to follow which reduces the greed to explore the unobserved space, since its reward is affected by deviation from the reference. IL methods associated with DRL have shown an efficient approach to enabling robots to learn skills from human operators. This line of research has been widely investigated and applied in different contexts.

XIE et al. (2019a, 2019b) proposed a control system for a stable bipedal walking of the Cassie robot by combining DRL policy-gradient and supervised learning methods in the MuJoCo simulated environment. For this, the proposed system is divided into three blocks. Block one consists of learning an initial policy with a DRL Actor-critic algorithm with motion tracking rewards. Block two iteratively improves the policy as needed with the estimated mixed policy gradient using PPO. Then, in block three, an SL algorithm combines and compresses multiple policies and distills them in a new policy π_n. The proposed supervised learning is named Deterministic Action Stochastic State (DASS). DASS is a set of (s,a)tuples collected from the reference policy, which allows to obtain a new policy from a small number of samples of the walking cycle motion. In conclusion, Xie et al. transferred the learned policy to the real robot achieving stable walking motions with different speeds.

GONÇALVES (2021) propose the utilization of DASS algorithm from XIE et al. (2019a) applied with SAC algorithm to train new policies from the imitation of the reference policy of walking gait proposed by CHENATTI et al. (2018a) to the Marta robot. Gonçalvez, utilizes the OpenAi gym interface for the DRL environment and PyRep for the CoppeliaSim. Also, develop a class for humanoid robots in PyRep.

In more recent research, researchers have focused on using only state demonstrations generated by an expert to teach an agent to imitate the desired task. This approach attempts to use an image or video references, where it is infeasible to obtain the action that describes the behavior. In the next section, we will address research that has used a human or robot character to mimic the reference motions of video demonstrations with DRL.

IfO applications in humanoid simulation

In this line of research, VONDRAK et al. (2012) is a pioneer work in achieving biped character control to mimic the reference of full-body 3D human motions from monocular videos in tasks such as walking, jumping, and gymnastics. To mimic the actor in videos, they employed a feedback mechanism of balance that maintains the balance of the body while producing the proper torque to guide the character towards a target pose, with the state-space controller based on constraints of the inverse dynamics actuation. They measured the probability relative to the difference of the 2D silhouette between the character and the reference. Recent approaches use the concept of IfO to train, with DRL algorithms, a character to imitate the reference motion from videos. MEREL et al. (2017) use GAIL as IL method and train with RL a character to reproduce a walking behavior in the MuJoCo simulator.

ZHANG; LIU; ZHOU (2019) proposed an IfO from a single video demonstration to train a character to mimic the reference skill. As shown the Figure 8, given a video input, the pose features are extracted by a supervised method. Then, in the feature matching module, a GAN is trained to learn the joint position controller that matches the expert position. The Discriminator module gives the probability to identify the similar trajectory motion between the controller and the expert. This function outputs a scalar reward signal to update the controller parameter in the Neural controller module. The policy is trained with the PPO algorithm. The MuJoCo environment is used for simulation.

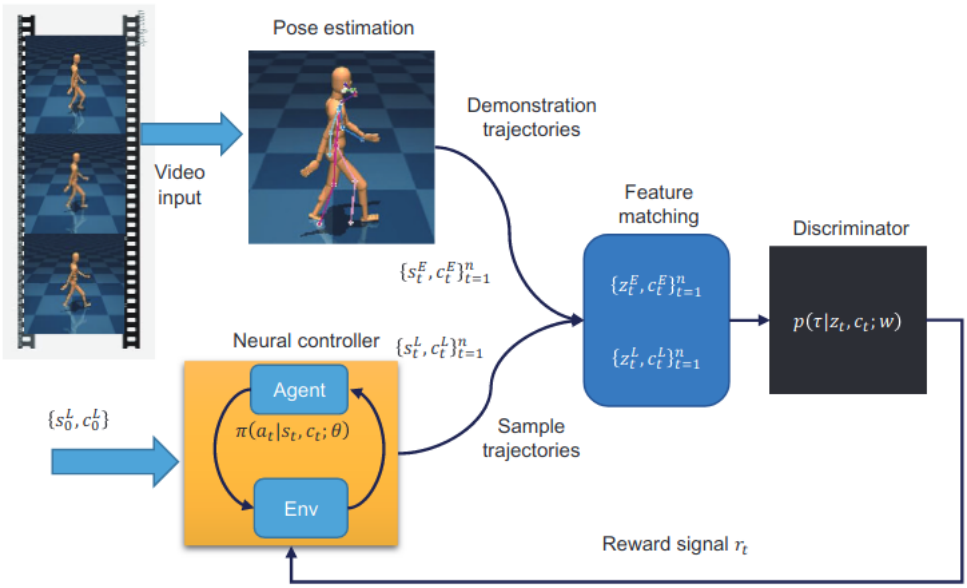


Figure 8 – Learning method proposed by Zhang, Liu and Zhou. Extracted from ZHANG; LIU; ZHOU (2019).

The complexity of video pose estimation can increase depending on the nature of skills such as dancing, acrobatics, and martial arts. In this context, PENG et al. (2018) proposed multi-skilled agents given the integration of multiple clips into the learning process and train the agents to perform an additional task such as throwing a ball to a target. The training process is started given a set of reference motions; each is a target pose. Then, the policy's goal is to reproduce the desired motion through an RL process. The policy action specifies each joint's target orientations for PD controllers since the reference motion only provides target poses. In addition to the preview work, PENG et al. (2018) adopt a motion reconstruction to accomplish the challenge from complex poses in YouTube videos. Both related works, utilizes the PPO DRL algorithm with Generalized Advantage Estimator (GAE) to compute the policy gradient.

YANG et al. (2020) proposed a imitation framework to train the simulated Valkyrie humanoid robot to walk. The imitation process used a reward engineering to extract the features of observed state from MoCap dataset videos with human performing the walking task. The authors train the robot in the PyBullet simulator, using the PPO DRL algorithm with GAE to compute advantages for policy updates during the train.

Synthesizing natural behavior for virtual characters and simulated robots is of great relevance for these fields. PENG et al. (2021) propose Adversarial Motion Priors, in which the choice of a given style carries a set of reference motions from video clips modeled as an adversarial discriminator between the reference to the simulated character behavior. The motion prior is incorporated as style-reward r_t^S for the policy during training, which measures the similarity between the reference to the character. This goal-conditioned reinforcement is increased with task-rewards r_t^G which encourage the achievement of a goal during the training policy (Figure 9). The approach does not require that character's behaviors exactly match the reference, but adopt high-level characteristics from the body's reference.

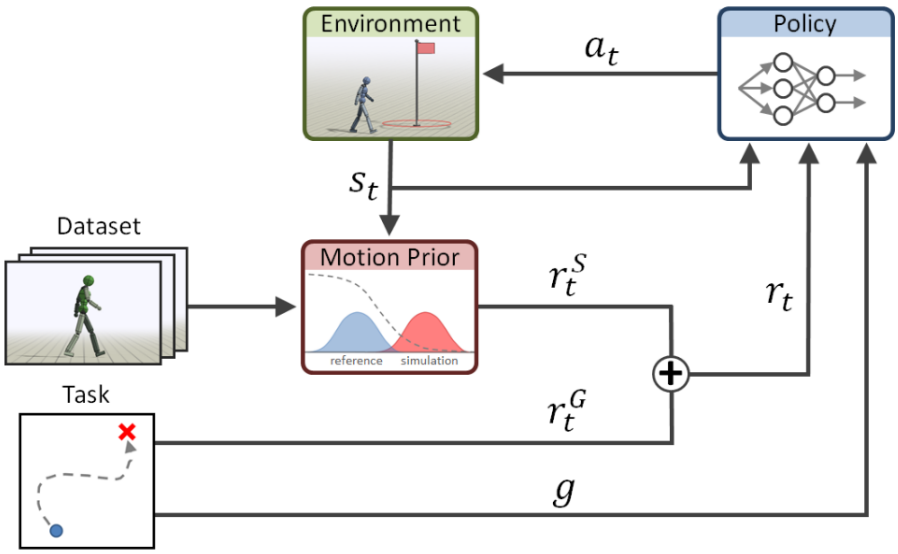


Figure 9 – Schematic overview of the Peng et al. work. Extracted from PENG et al. (2021).

However, addressing the body difference between the reference (human or animal) to the simulated imitator is barely addressed in previous works. In this context, HUDSON et al. (2021) adopt a skeletal feature compensation for imitation learning with embodiment mismatch, applying affine transformations to compensate the differences between the reference skeleton to the learner. This work applies GAIL to train a policy while learning a sequential affine transformation applied to skeleton features from the learner, given a loss function from conditional GANs.

SFV work review

PENG et al. (2018) approach is divided into three stages. First, given a video clip with a human performing a motion, the 3D pose estimation is applied, followed by the motion reconstruction stage for smoothing. Finally, given the reference motion, the policy π is trained with supervised learning to enable the character to mimic the reference motion using RL, as shown in Figure (10).

The Human Pose Estimation phase employs the framework Human Mesh Recovery (HMR) proposed by KANAZAWA et al. (2018). It estimates the 3D joint angles and employs the Skinned Multi-Person Linear Model (LOPER et al., 2015) body model to a 3D mesh of a human body, given the 2D pose estimation from OpenPose system. HMR applies a weakly-supervised adversarial approach with labeled poses to train the estimator to predict poses from a monocular image, which improved performance for acrobatic poses.

In the motion reconstruction phase, a new reference motion $\widehat{Q} = \{\widehat{q}_0, \widehat{q}_1, ..., \widehat{q}_t\}$, given the follow equation (PENG; KANAZAWA, 2018), is adopted:

$$min_{\widehat{Q}} = w_p l_p(\widehat{Q}) + w_{sm} l_{sm}(\widehat{Q}). \tag{17}$$

$l_p(\widehat{Q})$ is intended to encourage the new reference motion to behave similarly to the original pose predictions, and $l_{sm}(\widehat{Q})$ encourages the pose in adjacent frames to be similar in an attempt to achieve a smoother motion. In addition, w_p and w_{sm} are the weights for the losses.

The reference motion is an imitation goal to train the policy. The policy is training with a feedforward neural network, where the inputs are the current state s and the outputs are the action distribution $\pi(a|s)$. A variant of

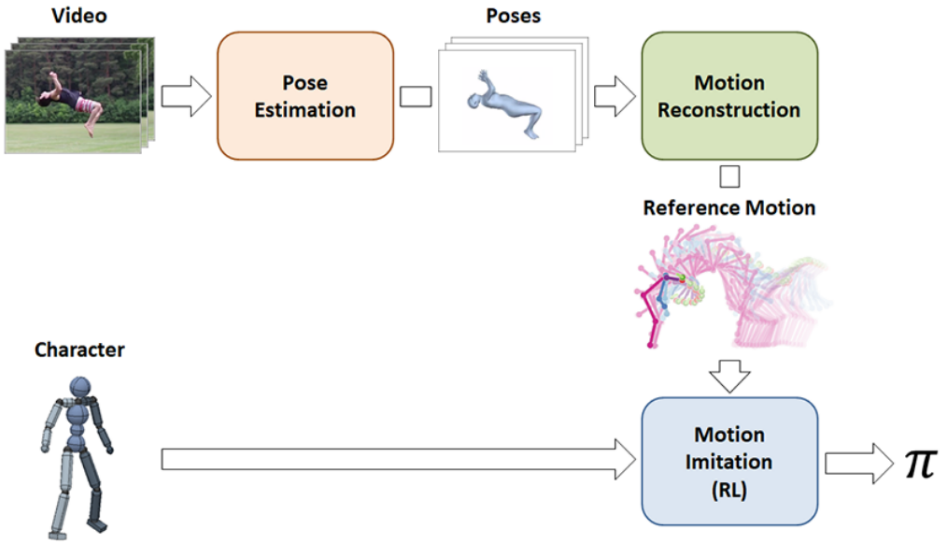


Figure 10 – Overview of proposed apporach by Peng et al. Extracted from PENG et al. (2018).

PPO with GAE to computed the advantages during the policy update. Xie et al. also adopt an adaptive initial state distribution to mitigate artifacts in the reference motion.

PENG et al. (2018) show that the initial state in which an agent begins each episode can significantly affect the learned policy quality, avoiding states that can be too complex for the character to recover from. Instead of the agent discovering by exploration the states that maximize the expected return, the proposal is to use the reference trajectory with promising states as a random distribution of state samples for the initial state.

The state s describes all features that affect the character behavior with the addition of phase variable by the measurement of the beginning and the end of the reference motion. The goal g is a feature for additional objective tasks, such as walking in some direction or hitting a target. Furthermore, the action specifies a target goal for PD controllers in each joint.

The reward r_t function is used to encourage the character to reach the reference motion. The reward function aims to minimize the least square error between the pose of the simulated character q_t and the pose of the reconstructed reference motion $\widehat{q}_t$ at each frame t, as shown in Eq. 18.

$$r_t = exp(-2\|\widehat{q}_t - q_t\|^2). \tag{18}$$

To reach the goal, different observations are considered, such as the measures of joint orientation from the reference motion, the difference between the local joint velocity, the adequacy of character hands and feet with reference motion, and the deviations in the character's COM from the reference motion.

In the training phase, the criteria for ending the episode are fixed or until an ending criterion is reached. The anticipated end occurs whenever the character falls, considered by touching the trunk on the floor of any agent's link, disabled for some tasks. As a result, Peng et al. showed the reproduction of twenty different dynamics and acrobatic skills of video clips collected from YouTube.

3.2 Pose estimation from videos with Deep Learning

Based on the number of persons in an image or video that HPE systems can perform, the approaches are called single or multi-person pose estimation. TOSHEV; SZEGEDY (2014) proposed seminal work that uses DNN to perform single-person pose estimation. GKIOXARI; TOSHEV; JAITLY (2016) adapt a sequence-to-sequence model for structured vision tasks. The input is a sequence of images in their work, and predictions are made at each step. At each step t, the model outputs the pose predictions for image X_t at that step using the past images and output variables.

The multi-person pose estimation presents a considerable challenge due to occlusion with other people or objects and the disappearing or reappearing of people. In this context, BERTASIUS et al. (2019) propose the Pose-warper network that feeds the CNN backbone with a pair of frames (labeled and unlabeled frame) from the same video to train the model and predict the pose heatmaps. The features of the unlabeled frame are used to train the model to detect the pose in the labeled frame and maximize its accuracy.

The entire sequence of frames in videos brings valuable information such as multiple views of a person, and the slow changes over time of the joint position in both 2D or 3D could help reduce the ambiguity in pose estimation (CHEN et al., 2020).

LIU; CHEN (2020) present a network architecture that in a range of temporal frames is processed in a single time. A pose propagation mechanism is applied, directly propagating the pose estimate through frames, transferring the labeled pose to adjacent unlabeled frames by building the unit's pose propagation networks. A similar approach, based on optical flow (movement between consecutive frames) for predicting heatmaps of adjacent video frames, is proposed by PFISTER; CHARLES; ZISSERMAN (2015) ZHANG et al. (2018).

ARNAB; DOERSCH; ZISSERMAN (2019) propose the 3D pose estimation models in monocular videos meshes through the packet adjustment method by exploring the temporal consistency between frames of a video to correct such ambiguity in the pose estimation. Li et al. (LI; YANG; LIAO, 2019) showed that capturing the temporal information also improves the accuracy of video pose estimation.

Recent approaches incorporate semi-SL and SL to train networks in pose estimation. LI et al. (2019b) proposed a two-phase structure to estimate a 3D human pose from a single image. In their work, training uses Semi-SL with few annotated data for initial predictions. It then estimates the pose from unannotated monocular video sequences. Semi-SL can be applied to solve the problem with a mismatch of labeled and unlabelled data. In addition, PAVLLO et al. (2019) show that Semi-SL can be applied when labeled data is limited.

3.3 Analysis

In this section we will briefly analyze the IfO reference works, to support the proposed approach selected for the development of this work that will be detailed in the Chapter 4. The summary table 1 presents the main characteristics of each IfO work described in this section. Each reference work uses an agent with different DoFs that vary depending on the simulator, e.g., the human character in MuJoCo has 56 DoF. In comparison, the human character in Bullet has 34 DoF.

The works 1,4,6 in the Table 1 only utilizes characters as agents in the proposed approaches. Works 2,3,5,7, faces the difficulties of transferring human movements to the robot, and the necessity of preprocessing the reference data or making adaptations, e.g., disabling Roll and Yaw joints of the robot. The robots used in these works are widely known in the literature. We propose extending this principles to the humanoid robot Marta, that has a distinct body scheme with its 25 DoF composed of revolute joints, still little known in the literature.

Besides the humanoid robot, we proposed generating a new dataset with videos from YouTube where human players perform soccer kicks. The kick movement is a challenging task given the dynamics of the movement that the player

executes to kick the ball. We propose for the robot Marta to achieve the imitation of the pose and perform the task of kicking the ball. In the previous works, only the work number 3 in the Table uses YouTube videos as references. This is more challenging approach due to the difficulties of finding good quality videos, and given the expensive HPE process to extract the reference Pose.

We proposed to use the robust CoppeliaSim simulator given the PyRep toolkit which is the python script interface to the simulator since python was chosen to develop this work.

Finally, the SAC algorithm was chosen rather than PPO because the entropy factor gives a more stable and less sensitive to hyperparameters than other algorithms in the training (DONG; DING; ZHANG, 2020). Also, the entropy regularization helps boosts the explorations during training, and consequently directly affects the reward function. And, HAARNOJA et al. (2018), shows that SAC is faster than PPO in the learning process. The proposed approach will be detailed in next section.

Nº	Reference	Agents	Method	Video Source	Simulator
1	(MEREL et al., 2017)	Human character	Train the policy with GAIL and TRPO	MoCap	MuJoCo
2	(PENG et al., 2018)	Human, T-Rex and Dragon characters, and Atlas robot	Train the policy with a variant of PPO with GAE. Adopt an adaptative initial state distribution and early termination, given a set of Multi-skill reference motion as imitation objective	MoCap	Bullet
3	(PENG et al., 2018)	Human character, and Atlas robot	Pose estimation with HMR system (KANAZAWA et al., 2018) with Motion Reconstruction; Train the policy with a variant of PPO with GAE. Adopt an adaptative initial state distribution and early termination.	YouTube	Bullet
4	(ZHANG; LIU; ZHOU, 2019)	Human character	Pose estimation with HMR system (KANAZAWA et al., 2018) ; Train the policy with GAN and PPO	Uninformed	MuJoCo
5	(YANG et al., 2020)	Valkyrie humanoid robot	Train the policy PPO DRL algorithm with GAE	MoCap	PyBullet
6	(PENG et al., 2021)	Human, T-Rex, and Dog characters	Train the policy with GAIL and PPO	MoCap	Bullet
7	(HUDSON et al., 2021)	Human, HalfCheetah, and Ant characters, and Atlas robot	Train the policy with GAIL and adopt sequential affine transformations	MoCap	Bullet

Table 1 – The work summarized here aims to train an agent to mimic the human motion through DRL from videos. The table shows the reference work and its agents, methods, video source, and the simulator used.

4 PROPOSED APPROACH

This section describes the proposed approach for the transference of the human movements from video to robots. Section 4.1 presents an overview of the proposed approach. Section 4.2 presents the 3D Pose estimation algorithm. Section 4.3 discusses the Reference Motion. Finnaly, section 4.4 presents the Motion Imitation process. Section 4.5 discusses the embodiment mismatch issue. Finnaly, section 4.6 addresses the transferring human movements problem as a RL problem.

4.1 Overview

Based on previous works described in section 3.1 (PENG et al., 2018; PENG et al., 2018), the proposed approach combines the idea of extracting human movements (the expert or reference policy) from videos, and imitating such movements in a humanoid robot. The imitation in this scenario refers to learning a new adapted policy, that is, adapting the expert policy to the structure and dynamics of the robot. Figure 11 presents an overview of the approach.

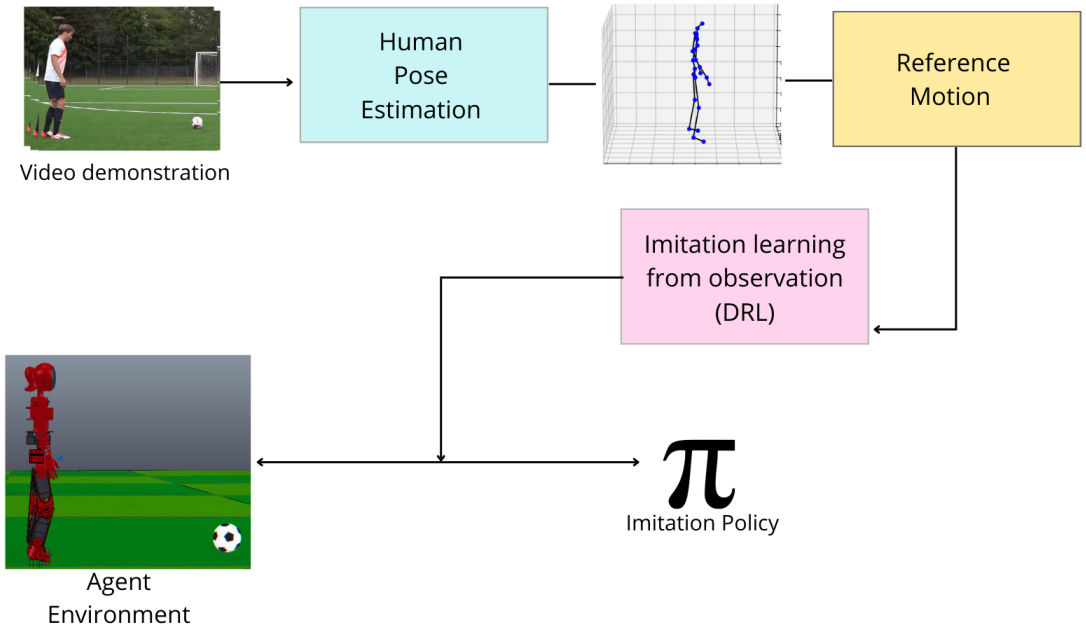


Figure 11 – Proposed approach overview of the three main phases of the process for imitating the movement of a Youtube video in the Marta: 1. Human Pose Estimation; 2. Reference motion and 3. Imitation learning from observation. The system output is the training policy π.

The proposed approach has three phases:

1. **Human Pose Estimation.** This phase consists on extracting a dataset of 3D human poses from videos collected from Youtube. This process is detailed in section 4.2;

2. **Reference motion.** The 3D human poses generated in the previous phase are in the camera's reference frame and the body shape is not necessarily compatible with Marta's. Therefore, we apply a set of transformations to obtain the reference frame corresponding to Marta's body. This process is detailed in section 4.3;

3. **Imitation learning from observation.** In this phase we apply imitation learning to mimic the reference movements adapting it to Marta's dynamics through DRL, while attempting to pursue an additional task. This process is detailed in section 4.4.

A detailed schematic representation of these processes is shown in Figure 12. The materials and methods adopted in each step of the proposed work will be detailed in next sections.

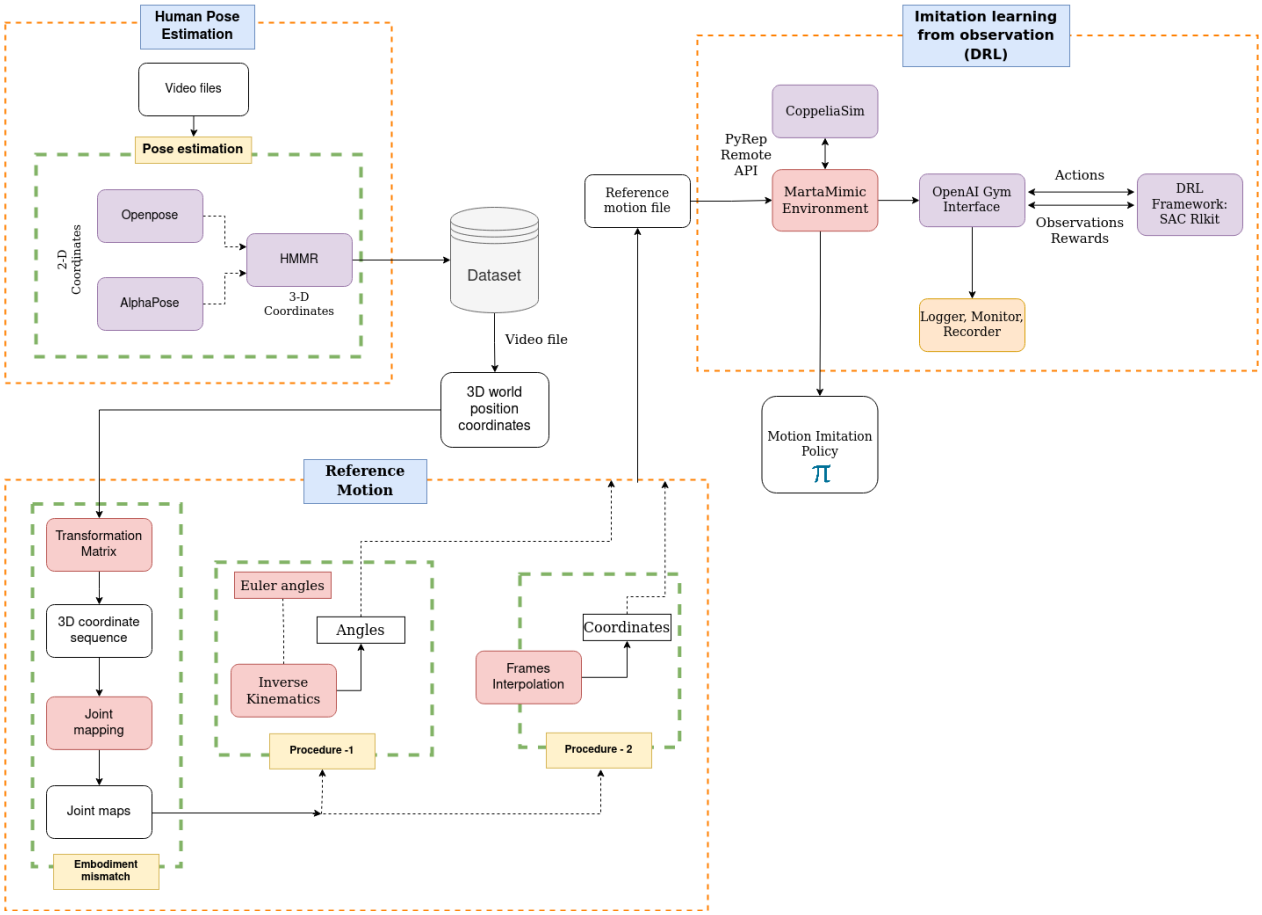


Figure 12 – Detailed schematic description of the three phases adopted for the imitation process.

4.2 3D Pose estimation

This section describes the creation of the *SoccerKicks dataset* (LESSA; COLOMBINI; SIMÕES, 2021a) (LESSA; COLOMBINI; SIMÕES, 2021b), a dataset of set-ball-kicks videos (penalty and foul) created by the author to provide reference motions suitable for humanoid robots.

4.2.1 The dataset

Considering the stupefying amount of videos available on social media, the first concern of this work is related to the organization of a movements dataset suitable for the proposed work. For creating the dataset, we performed an extensive manual search on YouTube database to collect a set of eligible videos, from distinct points of view (front, back and side), with players of different genders (male or female) and ages (adults and kids), following strength criteria for a successful Human Pose Estimation (HPE):

1. **Theme:** The selected videos refer to soccer games with the player performing two types of modalities in dead ball kicks: penalties or fouls;
2. **Action effectuation**: The selected videos contain a complete record of the player executing the following four steps during his/her kick motion:
    - Stage 0: The player is in standing posture and in front of the ball;
    - Stage 1: The player performs a quick thrust to start the movement;
    - Stage 2: The player kicks the target ball;
    - Stage 3: The player stands stably with both feet on the ground.
3. **Camera view:** No sudden movements (changes, shakes, rotations, or translations) should be present in the camera's point of view;
4. **Size and scale:** The player should have the whole body fit in the selected videos, and be focused on all the frames;
5. **Occlusion:** In the selected videos should not appear limb occlusions with objects or persons in the scene;
6. **Persons:** The selected videos should not contain many people in the scene or background, with the player as the main view;
7. **Quality:** The videos must have adequate contrast and lighting conditions, low noise or blurring that could diminish the quality of the videos.

4.2.2 3D Human Pose Estimation (HPE)

Given a selection of videos, the second step to construct this dataset involved the investigation of 3D HPE. We adopted the Human Mesh Motion Recovery (HMMR) system (KANAZAWA et al., 2019) that is able to estimate 3D poses based on 2D poses.

2D Estimation

Two distinct 2D HPE models (2D joint locations) were considered in the present work: the *AlphaPose* (FANG et al., 2017; XIU et al., 2018) and the *OpenPose* (CAO et al., 2017; CAO et al., 2019). Both systems are multi-person and real-time 2D pose estimator, but based on distinct approaches.

OpenPose (WEI et al., 2016; SIMON et al., 2017; CAO et al., 2017; CAO et al., 2019) can detect up to 135 keypoints in human body (hand, facial, and foot) on single images. OpenPose is a bottom-up approach where each joint is detected and associated with a whole person through Part Affinity Fields. A set of 2D confidence maps of each body part location and their part affinity fields are predicted by a feedforward network. Then, the sets are passed by

greedy inference to extract the 2D keypoints for each person in the image through bipartite matching. The confidence measures is adopted for associating which members belong to the same person, and is associated with PAFs to mitigate the limitations of false positives in multiple people's images. Each stage in this approach produces more refined predictions by chain of predictions from the early stages given the original image features. OpenPose was build on COCO and MPII datasets.

AlphaPose (FANG et al., 2017; LI et al., 2018; XIU et al., 2018) can detect up 136 keypoints by the pre-treined models COCO and Halpe. Alphapose is a top-down approach that lists all candidate locations to each joint, named joint-candidate Single Person Pose Estimation, that described if the joints belongs to the person under consideration (target joints) or if the joint belongs to other human instances (interference joints). The network use the loss function to set all the guess joints as candidates building a person-joint connection graph. Finally, a global maximum joints association algorithm is used to match the final joints with the persons. AlphaPose use the public datasets like MSCOCO, MPII, AI Challenger, and CrowdPose, to build the system.

The 2D *AlphaPose* was used with the pre-trained model Halpe (joint project under AlphaPose and HAKE) (LI et al., 2019a) with 26 keypoints annotations, and the 2D *OpenPose* detector system was used with its pre-trained model for 25 keypoints to estimate the 2D human pose. Since the videos can suffer interference due to the presence of people or objects, for example, the results of the 2D pose estimation algorithms were verified to assure the player were successfully detected in all frames with the minimum quantity of 20 detected keypoints (80% of the total).

The Pose Flow tracker (XIU et al., 2018) was applied to the *AlphaPose* results to sort the possible people into indices. Then, the person index is saved in a dictionary to check in each frame if some index appears or disappears during the video length. If it happens, the person's key is filled with None parameter and cut off at the end of the process. If the Pose Flow makes erroneously indexed other people in a scene with a single player, the raw data from AlphaPose is tested in the task of finding the person. In this case, the number of keypoints per frame is compared with the number of frames videos. If there is a match, this data is used. Otherwise, the video is dropped from the list.

In the *OpenPose* results, for each frame, the bounding box was calculated (center, scale, and rectangular box). Then, from one frame to the next, the Intersection-over-Union (ZHOU et al., 2019) was calculated to determine the degree of similarity between then and to find out to which person that current box belongs. In the first frame, the number of people is set as a reference to identify if any people appears or disappears during the video. If in the end there are no survivors, the video is dropped from the list.

For both approaches, the person or persons remaining are sorted from highest to lowest size data. Then, one person must be chosen to be used for the 3D HPE.

3D Estimation

The Human mesh and motion recovery (HMMR) approach proposed by Kanazawa et al. (KANAZAWA et al., 2019) can learn a representation of 3D dynamics of humans from video via a temporal encoding of image features. The 3D modules was trained to learn from pseudo-ground truth 2D pose annotations. Given the 2D keypoints, each frame is rescaled, cropped and centered on the detected person constituting a bounding box. Also, the Median and Gaussian filters are applied to reduced the noise. Then, the weak-perspective projection are predicted, formed by three camera parameters $T\ (s_f, t_x, t_y)$, which corresponding the s_f scale factor, and translation parameters t_x and t_y from the shift of the body with respect to the root (pelvis), and from the center of the resized image coordinate frame. In the sequence, their temporal information is encoded by several layers of a 1D fully convolutional network with a ResNet-50 (HE et al., 2016) based architecture. The visual features representation in the video clip encodes the 3D dynamics where the 3D mesh of the human body at frame t is predicted. The 3D mesh consists of 85 parameters (shape β, pose ϕ, and camera parameters Π) denoted by $\Phi = \{\beta, \phi, \Pi\}$, obtained using the SMPL body model. The SMPL body model can be

described by the model function $\mathscr{M}(\beta,\phi) \in \mathbb{R}^{N\times 3}$. Given $\beta \in \mathbb{R}^{10}$, a vector of shape parameters and, ϕ, the axis-angle representation relative rotation of body part k for the kinematic tree, the model outputs a vector with $N = 6890$ vertices of a triangular mesh (for $K = 23$ joints). The mesh has the same topology for men and women, defined by a standard skeletal rig that forms the body's pose. From the vertices and pre-trained linear regressor the 3D joints location are predicted, with 25 joints locations. Given the 3D joints' location and camera parameters, the 2D projective plane is mapped from 3D projective space. Finally, the mesh is rendered in the input video clip, and the mesh parameters are saved in a pickle file.

For the pseudo-ground truth 2D pose annotations, two datasets of videos from the internet were used: the VLOG-people (KANAZAWA et al., 2021b), a subset of the VLOG lifestyle dataset (FOUHEY et al., 2018), and InstaVariety (KANAZAWA et al., 2021a), from Instagram with a range of human dynamics categorized by 84 hashtags for actions. These datasets employed the OpenPose pre-treined in COCO to obtain the 2D pose prediction, with 25 joints locations. In addition, the MoCap Human3.6M, Penn Action, and a version of the NBA dataset were used to train full 3D supervision for each video with the ground truth 2D pose annotations. Finally, the approach was evaluated using the 3D Poses in the Wild (3DPW) dataset.

In the present work, we adopt different 2D HPE model and use the framework HMMR to estimate the 3D poses from both models. We adapt the Kanazawa et al. approach to use the 2D pose detector OpenPose system with their pre-trained model for 25 keypoints annotations, and adopt the AlphaPose with the pre-trained model Halpe with 26 keypoints annotations. Given the person 2D coordinates, the 3D HPE HMMR was applied to recover the 3D dynamics representation of the human from the video.

At the end of this process, as a preliminary implementation, a collection of selected videos were grouped into the *SoccerKicks dataset*.

4.3 Reference motion

The 3D human poses generated in the previous phase are in the camera's reference frame and the body shape is not necessarily compatible with Marta's. Some transformation is required to obtain the reference frame corresponding to Marta's body. The **Rigid Body Motion Transformation** describes the relation between the 3D coordinates point in the camera frame $P_c = (x_c, y_c, z_c)$, and the 3D point in object (world) coordinates frame $P_o = (X_o, Y_o, Z_o)$, for a **classical perspective camera model**. The rigid body transformation from the object frame to the camera frame is given by:

$$\begin{bmatrix} x_c \\ y_c \\ z_c \end{bmatrix} = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} \begin{bmatrix} X_o \\ Y_o \\ Z_o \end{bmatrix} + \begin{bmatrix} T_x \\ T_y \\ T_z \end{bmatrix}, \tag{19}$$

or equivalently,

$$P_c = RP_o + T. \tag{20}$$

Where R is the 3x3 rotation matrix and T is the translation vector, orientation and position respectively. Therefore, the pose of the camera describes the position of the moving object frame and the orientation of the body frame, concerning the camera frame (Figure 13).

Since the weak-perspective camera parameters predicted that the camera is too distant (unknown camera focal length f) from the object, this leads to the assumption that possible changes in the z coordinates of a point are insignificant compared to the distance from the camera (KISSOS et al., 2020). With this assumption, the camera translation

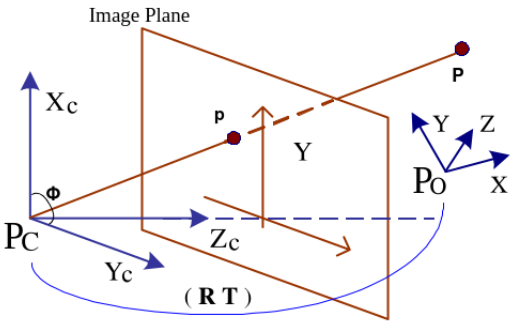


Figure 13 – The rotation ϕ about an axis and the translation from P_c to P_o in the camera. Adapted from WANG; WU (2011).

in the z axis t_z can be calculated by converting the scale parameters given a focal length approximation from image resolution:

$$t_z = \frac{2 \cdot f}{r \cdot Res \cdot s}. \tag{21}$$

The equation 21 considers *Res* as the resized box with a fixed resolution of 224 x 224, $r = b/Res$ where b is the size of the detected person in the bounding box, and assumes that the focal length is f = 500 pixels, such as proposed by Kissos et al. (KISSOS et al., 2020). Then, given the camera center shift parameters $(\widehat{c_x}, \widehat{c_y})$, the camera translation $\widehat{T} = (t_x - \widehat{c_x}, t_y - \widehat{c_y}, t_z) \in \mathbb{R}^3$ in the world coordinate frame can be approximated, assuming no radial or tangential distortion in the camera. From then, we can approximate the 3D joints locations predicted from the vertices and pre-trained linear regressor, with the 25 joints locations, to the absolute (global) 3D joints location. Furthermore, given the transformation matrix, we obtain the absolute (global) 3D joints location from the SMPL body model parameters with 24 points, where the first three parameters of ϕ corresponds to the global orientation of the body (root rotation), and the first tree parameter from the global joint coordinates are the global position (root position). The sequences of global 3D joints locations were grouped into the *SoccerKicks dataset*.

For this work, given the selected video from the dataset, we adopt the 3D joints locations from the SMPL body model with 24 joints locations. Then, transformations matrix is applied (such as rotations, scaled, translations) to obtain the coordinate sequences as described in the following processes.

Human Pose to Robot

In this work, the simulated version of the humanoid robot Marta, described in section 4.5.1, will be adopted. Marta, as shown in Figure 16, is a female robot with 25 active revolute joints (DoFs) which allows rotational motion about their joint axis, and with 3 active revolute joints that describes the motion of a spherical joint at the waist (Hip yaw, Hip pitch, and Hip roll).

On the other hand, the movement of the human body is described according to the type of joint that connects the limbs. The human shoulder and Hip has the called Ball-and-socket joints that allow a 3D movement, and is described as spherical joint. The elbows and knees joints is called hinge joints allow a 1D movement backward-forward swing (MACCONAILL, 2021). The Figure 14 is an illustration of the human joints in motion.

Given the human body motion as reference movement, and considering the robot Marta joints configuration, in case of the 1D revolute joints available at Marta's body, we have the problem of embodiment mismatch.

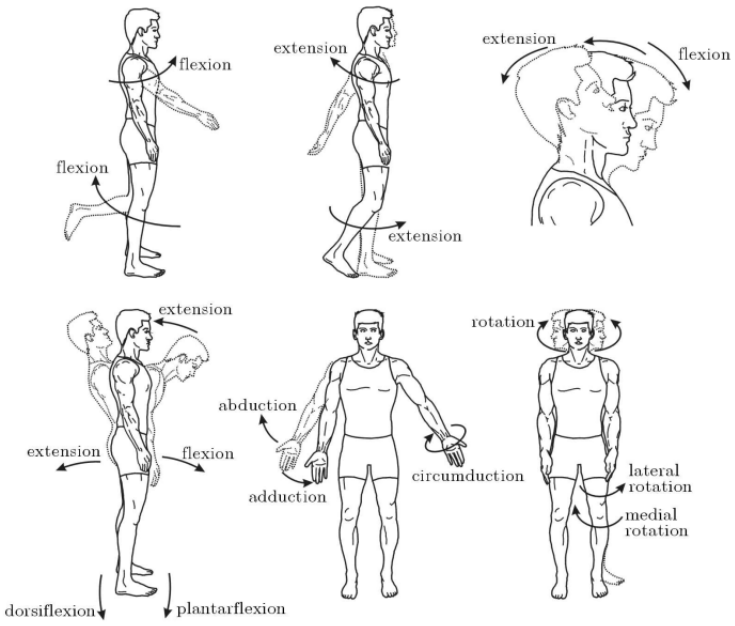


Figure 14 – Representation of human joints in motion. Extracted from GŁOWIŃSKI; KRZYŻYŃSKI (2016).

Since our goal is to build a framework to train the robot to learning from states observations from human reference, to address the inherent problem of embodiment mismatch, and considering that we will adopt the reward engineering as the control component, we adopt two methods that we call *Procedure* – 1 and *Procedure* – 2. Both methods are described in the Chapter 4.5.

- *Procedure* – 1: consist of application of Inverse Kinematics on the 3D coordinates position from video demonstration to extract the set of rotations from the mapping of each joint existing in both the reference body and the robot body. The set of rotations is then used as expert reference in the reward engineering in the DRL process. This method was based in the related work described in 3.1.

- *Procedure* – 2: is the use of rescaled (to the robot height) 3D absolute coordinates position from video demonstration as reference. An interpolation function as an intermediate process is also adopted to smooth the coordinates values.

For the *Procedure* – 1, the source 3D coordinates positions that describes the motions from a given video, is displaced in time and starting with the pelvis aligned at zero. From them, a joint mapping is made in the reference skeleton (Figure 15 a.) to find the target joints correspondents in the robot's body for IK calculation.

From the joint mapping, we adopt the following target joints, for the right and left sides, to extract the set of rotations: shoulders, elbows, chest, pelvis, leg, and knees. The shoulders, chest, pelvis, and leg were modeled as 3D joints. The elbow and knee, in turn, describe a uni-dimensional movement modeled as a 1D joint. From the joint map, the IK calculation was based on the work published in (PYBULLET, 2019).

The matrix of rotation for the 3D joints were extracted from the group of related joints, as shows Figure 15 b.). Therefore, the first step was calculate the pelvis rotation as origin of the chain of rotations. The pelvis rotation matrix is a result of rigid transformation matrix of the cross product of the trunk joints (chest,spines,and pelvis) with the middle point (simple average between the legs joints positions), and leg side. From then, we can calculate the chest and leg

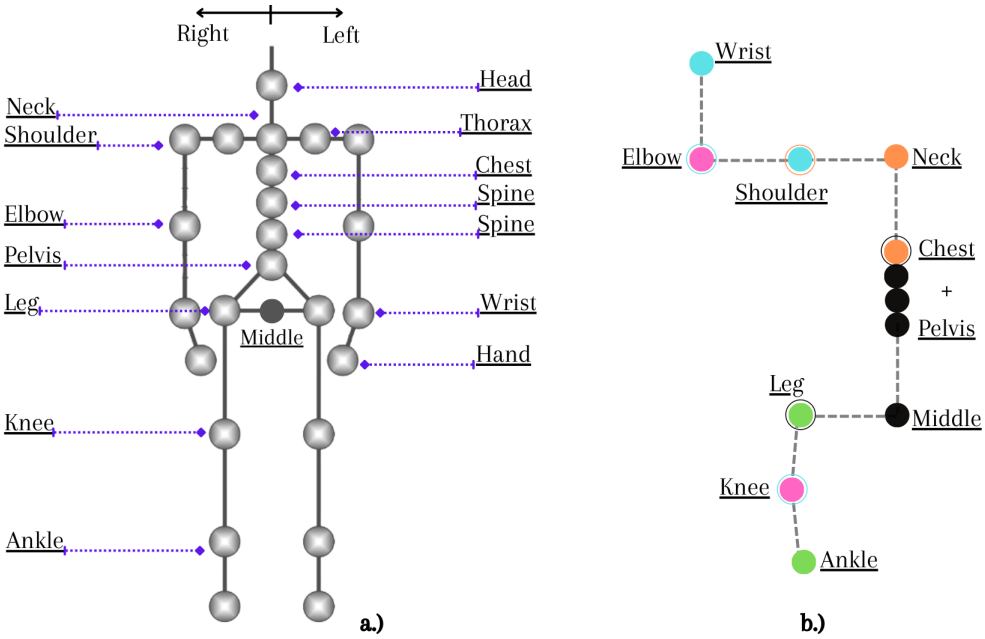


Figure 15 – The Figure a.) is the skeleton representation of SMPL body model with 24 points localization. The Figure b.) is the joint map used to compute the IK.

rotations with the root rotation as related orientation. The shoulders rotation is related to the chest orientation. The Elbows and Knee scalar rotation angles is calculated with a simple IK for revolute joints. The ankle joints were not calculated based on IK, because the humans always adjust the angle of the ankle joint to preserve the body balance. Instead of that, the position coordinates were considered to elaborate the reward engineering.

In Figure 15 b.), the colors identifies the related triangular joints for each rotation. The joints elbow, shoulder, chest, leg, and knee are points used for their own and related IK calculations. For the calculation of the shoulders rotation, the elbow, wrist and shoulder are related points, for example. The final rotations matrix, for each spherical joint, is transformed into the form of 4D quaternion rotations. The set of rotations is saved in a JSON file.

With the set of rotations for each target joint in the human reference, we apply the quaternion transformations for Euler angles on the 3D spherical joints to transfer them to the 1D revolute joints on the robot body. This process is detailed in Chapter 4.5.

4.4 Robot learning from motion imitation

Since we have a valid reference motion, the last phase is the motion imitation, which aims to enable the robot to reproduce the demonstrated movement by learning a policy π with RL. The goal of RL is to maximize the expected return by finding a policy π, parameterized by θ. The agent interacts with its environment described by the policy $\pi_\theta(a|s)$, that models the conditional probability distribution $p_{\pi\theta}(a|s)$, given the state s_t and action a_t at time t, characterized as an MDP. Then, solving the optmization problem by $max_\theta J_{rl}(\theta) = E_{p_{\pi\theta}}[\sum_{t=0}^{T} \gamma^t r(s,a)_t]$, given the discounted return of a trajectory $\sum_{t=0}^{T} \gamma^t r(s,a)_t$ in T steps with a discount factor γ. Therefore, given an expert reference motion, the proposal is to find a policy π_θ which minimizes the divergence between the distribution of learner-induced features (simulated robot pose) q_t, and the features induced by expert policy (reference motion) $\widehat{q}_t$ in each frame at time t, as described in

Eq. 22.

$$\pi_\theta = \arg\min \|\widehat{q}_t, q_t\|. \tag{22}$$

Also, we set an additional goal of kicking a ball. For this purpose, the DRL method with Soft Actor-Critic algorithm (SAC) will be applied for training a policy π_θ. The action specifies a target position in each joint. The reward r will encourage the robot to minimize the error between its pose with the reference motion. A detailed explanation about the RL modeling is present in section 4.6.

4.5 Embodiment mismatch

As discussed in Section 2.4, one of the main components of IfO method is the Perception. From this, this section will address the Embodiment mismatch between the reference to the robot's body.

4.5.1 The Humanoid Robot Marta

The humanoid robot Marta was developed by the working group: *Grupo de Automação e Sistemas Integrados* (GASI) at the Institute of Science and Technology of Sorocaba, *Universidade Estadual Paulista* (Unesp) and *Laboratório de Robótica e Sistemas Cognitivos* (LaRoCS) at *Universidade Estadual de Campinas* (Unicamp) (CHENATTI et al., 2018b). Marta, shown in Figure 16, has 25 DoFs, with three revolute joints at the waist (Hip Yaw, Hip Pitch and Hip Roll), allowing a more realistic movement for the humanoid robot with the movements around the 3 axis. Other relevant characteristics of the robot are the small contact area on the foot and the toe joint, which gives an extra degree of freedom on the foot.

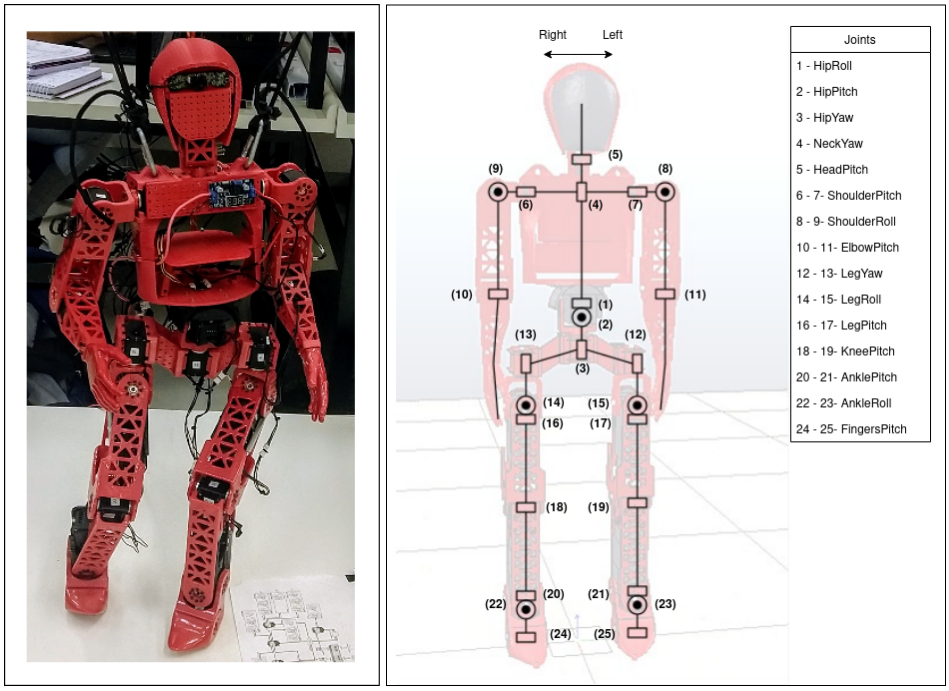


Figure 16 – Marta humanoid robot. a) Physical version; b) Highlight of the 25 robot joints, described in the left corner.

The list of the limits rotations in degrees for each Marta's joint is presented in Table 2.

Joints Names	Rotation Limits in Degrees (°)
LeftPelvisPitch	-40.7 to 85.5
RightPelvisPitch	-48.31 to 96.39
LeftPelvisRoll	-15.94 to 72.03
RightPelvisRoll	-18.86 to 49.15
LeftKneePitch	-87.68 to 69.2
RightKneePitch	-87.68 to 69.2
LeftAnkleRoll	-28.62 to 57.43
RightAnkleRoll	-36.72 to 77.31
LeftAnklePitch	-12.13 to 68.28
RightAnklePitch	-11.17 to 56.51
LeftFingersPitch	-27.16 to 72.62
RightFingersPitch	-42.67 to 75.32
LeftShoulderPitch	-45.0 to 90.0
RightShoulderPitch	-45.0 to 90.0
LeftShoulderRoll	-30.0 to 35.0
RightShoulderRoll	-30.0 to 35.0
LeftElbowPitch	-123.9 to 131.5
RightElbowPitch	-118.2 to 123.5
HipRoll	-49.1 to 99.4
HipPitch	-30.5 to 55.2
HipYaw	-25.16 to 43.98
LeftLegYaw	-73.57 to 130.4
RightLegYaw	-48.2 to 100.0
NeckYaw	-180.0 to 180.0
HeadPitch	-180.0 to 180.0

Table 2 – Marta's joints Rotation Limits in Degrees. Extracted from GONÇALVES (2021)

Marta is about 1m height and weights 8Kg. It was built using 3D printing technique and ABS plastic. Two Lipo batteries of 14.8V and 4400mAh feed the processor, actuators and sensors. The robot uses the Dynamixel AX-12, with stop torque at 1.5 N.m, and Dynamixel MX-64T motors, with the maximum stop torque at 7.3 N.m. The Dynamixel MX-64 is used in the twenty-three (23) highlighted joints. The *HeadPitch* and *NeckYaw* uses Dynamixel AX-12 motors. The robot has several sensors, like camera, orientation and pressure sensors. The UM7 orientation sensor (REDSHIFT-LABS,) contains a three-axis accelerometer, rate gyro, and magnetometer data. In each foot the robot has seven flexible piezoresistive force sensor, the A207 FlexiForce model (TEKSCAN,). The main controller is an Intel NUC Mini-PC *ultratop brix*, with an Intel Core i7 processor, that allows the operation of external data that is received from the motors and sensors.

4.5.2 Embodiment mismatch approaches

To address this problem, we adopt two methods to construct reward engineering. The first method ($Procedure-1$) is the application of IK in the 3D position coordinates to extract the set of rotations for each target joint. The second method ($Procedure-2$) is using the rescaled (to the robot height) 3D coordinates position as a reference. We also investigate for the second method (Procedure -2) the application of interpolation function, for each frame reference, for smooth motion transitions, detailed in Section 4.5.2.

Motion Retargeting: *Procedure* – 1

The source 3D position coordinates are retargeted to the robot's morphology with IK, described in Section 4.3. From the joint mapping of the human reference skeleton (Figure 17 a.), the target joints correspondents in the robot's body are shown in the Figure 17 b.). We map thirteen target joints (Right and left joints) in the human reference skeleton. From them, the target human body parts shoulders, chest, pelvis-hip, and pelvis-leg describe a 3D movement that we transform the rotation of quaternions to Euler angles (α, β, and γ) for each correspondent revolute joints direction after the IK. Elbows and Knees naturally describe a one-dimensional motion.

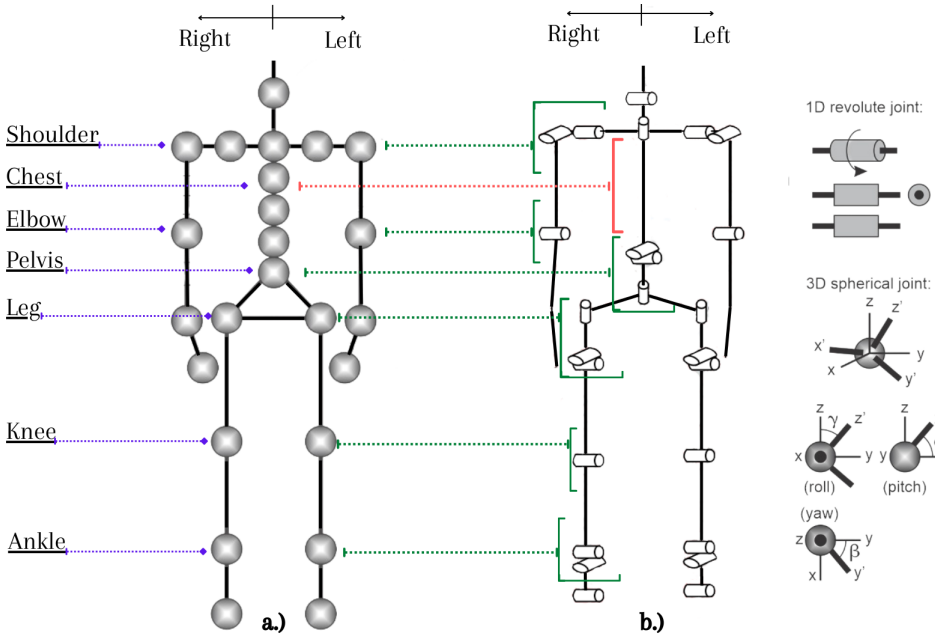


Figure 17 – Figure a.) is the SMPL Skeleton with the target joints in evidence. Figure b.) The green lines connect the target joints of the Marta robot with the joint group of the reference. The orange line indicates that from the chest of the reference we evaluate its orientation to the robot's chest. The figures are illustrations of each skeleton reference, so they are not represented in real scale.

To minimize the error between the root and ankle positions of the robot to the reference, a height offset is calculated at the restart of each episode, which is added for each frame at time t. At the beginning of each episode, an offset position and orientation are applied, so that the pelvis of the reference is aligned with the robot.

Motion Retargeting: *Procedure* – 2

Given the 3D absolute coordinates of reference, we rescale to the robot's height. Since we have two different body shapes, the localization of each joint differs between them. Therefore, a permanent error coexists in the imitation learning process. To minimize the error at the beginning of each episode, we align the pelvis of reference to the robot start position and apply the offset for each joint. We expect the robot follows the trajectory of the reference positions.

Frame Interpolation

To the effect of motion retargeting, given the complex motions, we also study the adoption of an Interpolation function to produce non-existent intermediate frames of two adjacent frames. The interpolation function is an efficient way to

obtain a smoother and more fluid set of rotations to then apply to the robot's joints. Based on the work of PENG et al. 2020b, we apply the interpolation function in a given point in time video duration. For each step of the simulation, the time step video is incremented, and then the interpolation function gets the adjacent frames and extracts a frame fraction (dt_{blend}) between them. The frame fraction is calculated as the Equation 23 shown. Given the current point normalization (p_{norm}), which is the multiplication of the time cycle of the motion with the current phase of the motion point time (scalar value between 0 - first frame, and 1 - last frame), and the adjacent frames time ($frame_{tn}$ and $frame_{tn+1}$.

$$dt_{blend} = \frac{(p_{norm} - frame_{tn})}{frame_{tn+1} - frame_{tn}} \tag{23}$$

For each joint position and velocity a simplified function is applied for all the target joint in the reference coordinates, as Equation 24. Where $Frame0_j p_{nt}$ is the current frame of positions, and $Frame1_j p_{nt}$ is the next frame of positions in given point time.

$$target\,joint = (1.0 - dt_{blend}) * Frame0_j p_{nt}) + dt_{blend} * Frame1_j p_{nt}. \tag{24}$$

Similarly, the interpolation of the pose velocities is calculated under the joint type. In addition, to keep the reference's pelvis aligned with the robot after a given number of motion cycles, an offset position for the X and Y axes are added for each frame at time t.

4.6 Transferring Human Movements as a Reinforcement Learning problem

As discussed in Section 2.1, an RL problem can be described as an MDP, a general mathematical model for modeling control problems. In the particular case of motion transferring, successful results were obtained in previous work, detailed in chapter 3. This chapter will address the essential part of the imitation task, which is the control method that will be used. The imitation task will be addressed by the SAC algorithm, as described in Section 4.4.

4.6.1 Reinforcement Learning problem

Transferring human movements as a reinforcement learning problem framework was designed by the careful analysis of objectives versus the materials chosen and available, detailed in Chapter 5. To model the motion imitation problem and task objective as an RL problem, first is necessary to unravel and shape the RL architecture, formed essentially by the observation space, action space, and reward function. The state and state transitions probability require a minor additional configuration once it is implicit from the physics engine simulation. In the physics engine simulator the gravity is maintained constant in $9.8\frac{m}{s^2}$ at each time step dt. To approach the motion imitation RL problem for the two methods (Procedure -1 and Procedure -2), the MartaMimic environment is modeled as described in the following sections. The motion cycle is repeated during the whole time of the episode to minimize the simulator delays.

4.6.1.1 Initial state distribution

We random sample the initial state distribution from the reference trajectory. Previous works presented this strategy (PENG et al., 2018; PENG et al., 2018; PENG et al., 2020a), which have demonstrated efficient results using this approach.

4.6.1.2 Observation Space

The observation space is formed by all the important information that we consider relevant for the agent to learn to imitate the pose and kick the ball. The observation space must be carefully formulated, avoiding redundant or extra information to maintain the learning process's efficiency. In our problem, the observation space is composed of Proprioception observation inputs, Sensorial observation inputs, and Additional observation inputs.

Proprioception observation inputs

The Proprioception is all the information relative to each part of Marta's body's positions, orientations, and velocities. The observation state adds all the 25 revolute joints rotations angle and angular velocities. The following specific information is also added to the vector.

- Chest Coordinates and Velocities: the chest Z-axis position, rotations (α, β, γ), and linear and angular velocities are added. The Chest information brings valuable information about the upper body stabilization following the reference movement;
- Pelvis-Hip Coordinates and Velocities: the Hip Yaw revolute joint coordinates (x,y, and z-axis) and its linear velocity. The Hip position is evaluated in the reward engineering to follow the pelvis position of the reference;
- Z-axis Position for each part shape: the Z-axis position for each part shape of Marta's body, which is measured as a condition for the episodes to terminate;
- Ankles Coordinates: the ankle's pitch coordinates position (x, y, and z). The ankle joints position is also evaluated to follow the reference trajectory;
- Previous Action: the previous actions (angular positions for all joints) taken by the agent.
- Joints positions coordinate: exclusively for training the Procedure -2 method. We added the 3-axis absolute position coordinates for each of the 25 revolute joints.

Sensorial observation inputs

The robot's perception of the environment is critical to interpreting the observational space. The sensory information added to the observation vector is the pressure force sensors, accelerometers reading, Center of Mass and Inertia. All this sensory information is fundamental for the stability and balance of the robot.

- Pressure Force Sensors: the four, two for each foot of Marta, pressure force sensors are read. For each sensor, we consider the x, y, and z-axis;
- Accelerometers: the accelerometers located in Marta's chest are read, and all the x, y, and z-axis were added. This measurement provides valuable information about the stability of the upper body and its orientation;
- CoM and Inertia: the relative mass, CoM position (x,y, and z), and Inertia matrix of the Marta are attached to the observation vector. This data provides information about the body's resistance to rotational acceleration about a rotational axis and then the required torque for a desired angular acceleration, accounting for all external forces acting on the system. The localization of the CoM has a critical effect on the robot's stability.

Additional observation inputs

One of the robot's goals besides the pose imitation is to hit the target Ball. The Ball is located at a distance d_{max} to the Kick foot located in P_0. An example is presented in Figure 18.

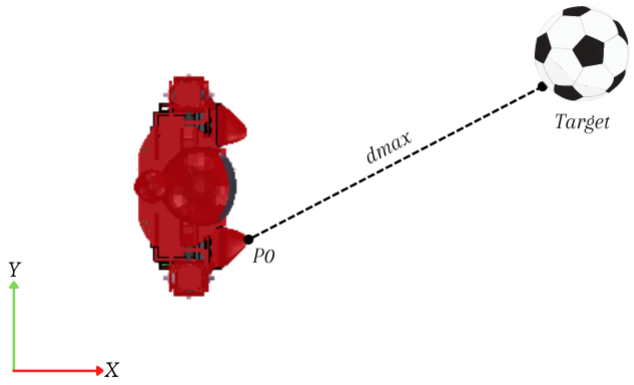


Figure 18 – Example of the task objective. The ball is the target in a d_{max} distance between the kicking foot to the ball-target. P_0 is the initial position of Marta's foot in the scene. This diagram is not in scale.

To evaluate this in the NNs, we added the state vector for the chosen Marta's kicking foot to the ball that follows the Finger shape positions (x, y, and z); the minimum distance between the foot to the ball; the state of collision, and the hit power. The Ball position, orientation, and linear and angular velocities are also part of state information.

4.6.1.3 Action Space

For our problem, we decided to use joint position control. As previously mentioned, all 25 activated joints in Marta are revolute. We treated the hip's spherical joint as three revolute joints. All joints have upper and lower angular limits (Table 2), which the agent must learn to manage.

4.6.1.4 Reward Function

The Reward function is designed to encourage the robot Marta to match the sequence of target poses $\widehat{q}_{t_n}$ at each time step while trying to reach an additional goal such as kicking a ball. For this purpose, we follow a similar approach used by Peng et al.(PENG et al., 2018; PENG et al., 2018) for training the policies with the Procedure -1. For training with the Procedure -2 method, we had to modify the structure but maintain the main characteristics. Therefore, the reward function is a weighted combination of multiples returns consisting of 6 main functions in total, each describing certain characteristics of the reference motion and task objective. The 6 functions are: Root Reward, Pose Reward, Pose Velocity Reward, End-Effect Reward, CoM Velocity Reward, and Task Reward. The Equations 25, 26, 27, 28, 29, and 30 describes the weighted sum of each of the components that make up the learning goals.

$$Reward+ = W_{Pose} * R_{Pose} \tag{25}$$

$$Reward+ = W_{Root} * R_{Root} \tag{26}$$

$$Reward+ = W_{Vel} * R_{Vel} \tag{27}$$

$$Reward+ = W_{EE} * R_{EE} \tag{28}$$

$$Reward+ = W_{CoM} * R_{CoM} \tag{29}$$

$$Reward+ = W_{Goal} * R_{Goal} \tag{30}$$

Each reward associated is described as an exponential function with a negative scalar factor, that multiplies the imitation error function. Each function and the associated exponential function are described for each adopted training method in the next sections.

Reward Function - Procedure -1

For the Procedure -1, each reward function is composed of the following: a set of orientations from a pose reward, pelvis position, orientation, angular and linear velocities reward, the pose velocities reward, the end-effector reward measured by the ankle positions, the velocity deviations in the robot's center of mass from the reference motion, and objective task reward.

1. Root Reward (R_{Root}):

   The root error is a weighted sum of each of the following errors between the robot's hip (Hip Yaw joint) to the reference Human pelvis-hip: root rotation (Equation 31), position (Equation 34), angular (Equation 32) and linear velocities (Equation 33).

$$root_{error}+ = 0.1 * \|\widehat{q}_{root_t} \ominus q_{root_t}\|^2. \tag{31}$$

$$root_{error}+ = 0.01 * \|\widehat{q}_{root_{vt}} \ominus q_{root_{vt}}\|^2. \tag{32}$$

$$root_{error}+ = 0.001 * \|\widehat{v}_{root_t} - v_{root_t}\|^2. \tag{33}$$

$$root_{error}+ = \|\widehat{p}_{root_t} - p_{root_t}\|^2. \tag{34}$$

$$R_{Root} = exp[-Root_S(\sum_j \|root_{error}\|)]. \tag{35}$$

   The root pose has an essential part of the reward function since it is responsible for giving the robot incentives for stability.

2. Pose Reward ($Pose_R$):

   The Pose reward attempts to encourage the robot to match the sequence of target joints rotation from the reference motion. Given a set of 1D joints rotations from the reference motion for each frame, we compute the error between the reference to the robot target joints, described in 4.3. Also, the Chest orientation is evaluated. The robot joints that will be evaluated in this phase are training parameters that can be rearranged as needed. Each target joint has an associated weight factor that describes the Pose error as the Equation 36 shown.

$$pose_{error} += J_{wn} * \|\widehat{q}_{jnt} \ominus q_{jnt}\|^2. \tag{36}$$

   The associated weight factor adjusts the measure of importance for each target joint.

$$R_{Pose} = exp[-Pose_S(\sum_j \|pose_{error}\|)]. \tag{37}$$

3. Pose Velocities Reward (R_{Vel}):

   Similarly, the pose angular velocities reward is calculated given the error of the target pose velocities (Equation 38).

$$Vel - error += J_{wn} * \|\widehat{q}_{vjnt} \ominus q_{vjnt}\|^2. \tag{38}$$

$$R_{Vel} = exp[-V_S(\sum_j \|Vel_{error}\|)]. \tag{39}$$

4. End-Effector Reward (R_{EE}):

   We assess the end-effector reward by the ankle coordinates positions after the offsets, which attempt to incentive the robot to follow the reference position. The end-effector position reward is an incremental incentive to arrive at the time-shifted positions. It is an ally to reaching the target task.

$$End_{error} = \|\widehat{p}_t - p_t\|^2. \tag{40}$$

$$R_{EE} = exp[-End_S(\sum_j \|End_{error}\|)]. \tag{41}$$

5. Center of Mass Velocity Reward (CoM_R):

   The CoM linear velocity error drives the robot to follow the reference CoM velocity and also contributes to the robot stability incentives.

$$com_{error} = \|\widehat{v}_t - v_t\|^2. \tag{42}$$

$$R_{CoM} = exp[-CoM_S(\sum_j \|com_{error}\|)]. \tag{43}$$

6. Goal Reward (R_{Goal}):

   The goal (or task) reward is measured by the error between the distance of the kicking foot to the start ball position.

$$task_{error} = \|\widehat{p}_t - p_t\|^2. \tag{44}$$

$$R_{Goal} = exp[-Task_S(\sum_j \|task_{error}\|)]. \tag{45}$$

   If the robot hits the target, the reward received is one until the end of the episode.

Reward Function- Procedure -2

For the Procedure -2, each of the 6 Reward functions is composed of the following: a set of position coordinates of each target joint form the Pose Reward, for each target joint, the linear velocities form the Pose Velocities Reward, the pelvis position and linear velocities form the Root Reward, the Ankle positions is the End-Effector Reward, the velocity deviations in the robot's center of mass from the reference motion, and objective task reward. Therefore, from the equations described for the Procedure -1, the modifications rely on the 1,2 and 3 functions.

4.6.1.5 Early Termination

Since our problem is to create a robust and effective motion imitation, keeping every part of Marta's body according to the reference in a stable manner throughout the movement imitation is a challenge. Therefore, as episode termination criteria, we analyzed the z coordinate of each Marta's shape, defined as an input parameter as forbidden to touch the ground. When its defined part touches the ground, Marta's is considerable as fall. Therefore, if Marta's z body part coordinate is below the defined number during an episode, Marta will or has already lost its stability. Thus, we must end the episode.

5 MATERIALS AND METHODS

This chapter describes the materials and methods adopted in the experimental process. Section 5.1 addresses the SoccerKicks dataset, proposed in the context of the current work. Section 5.2 presents the simulation platforms adopted. Finally, section 5.3 discusses the experimental procedure.

5.1 SoccerKicks Dataset

This section presents the *SoccerKicks dataset*, a dataset that was generated as part of the current work and that will be adopted in next sections. This section comprises: *i)* the methodology to choose the videos, and *ii)* the HPE from the videos selected to generate the reference motion. A more detailed description of the dataset is available in (LESSA; COLOMBINI; SIMÕES, 2021a). The database is publicly available at (LESSA; COLOMBINI; SIMÕES, 2021b) [1].

5.1.1 Dataset selected videos

The SoccerKics dataset has 38 videos with 2D and 3D pose annotations, detailed in the section 4.2.1. The selected videos are divided into two categories: *Penalty Kick* and *Free kick*. After processing, 23 videos were able to recover using one of the two 2D HPEs analyzed (10 videos for *Penalty kick* and 13 videos for *Free kick*), and other 15 videos (6 *Penalty Kicks* and 9 *Free kicks*) successfully passed in both 2D HPEs. They were evaluated using the average $l2$ norm and PCK metrics. The dataset contains 27 videos with men, 8 videos with woman and 3 videos with child players. The frame 0 of each video of the dataset is shown in Figure 19.

Figure 19 – SoccerKicks dataset overview. The dataset contains 38 videos distributed among the categories: Penalty and Free kick.

[1] SoccerKicks dataset GitHub page: <https://github.com/larocs/SoccerKicks>.

5.1.2 HPE evaluation

The dataset videos successfully achieved at least 80% performance in the joints identification in all frames using the 2D HPEs. Figure 20 presents a sample video correctly processed by both HPE systems. The difference is subtle between both algorithms in this case.

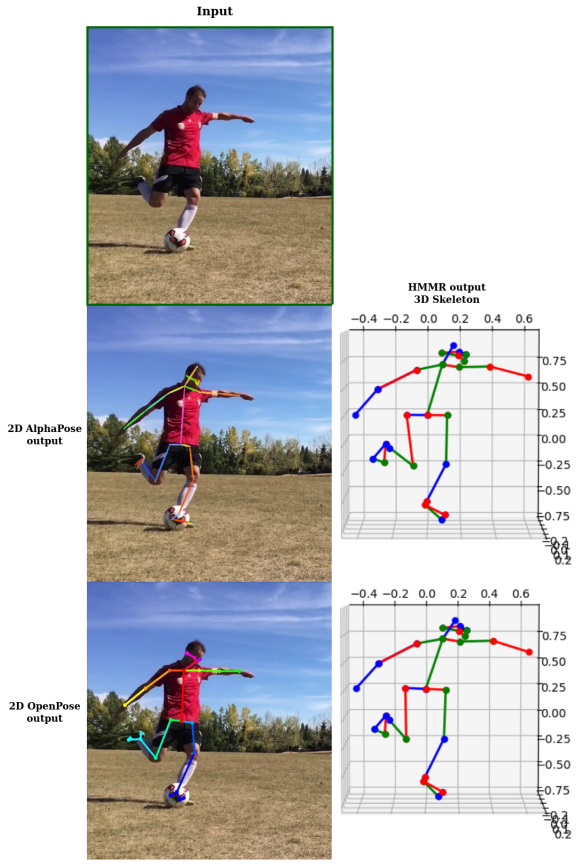


Figure 20 – Sample video successfully recognized by both 2D pose estimators. From top to bottom: a) First frame of the original video ($t = 0$) with a man in kicking pose; b) Original image rendered using AlphaPose 2D HPE and the resulting 3D HPE skeleton (right); c) Original image rendered using OpenPose 2D HPE and the resulting 3D HPE skeleton (right). The difference between the second and third frame is subtle given the compliance with selection criteria

Figures 21 and 22 presents examples of fail detections. The typical causes for 3D recognition fail are the poor image quality, low visibility, partial or total limb occlusion caused by body parts, other people or ball. Figure 23 shows the comparative results of the 15 videos that successfully passed both AlphaPose and OpenPose algorithms, given the $l2-$norm computed for both 2D HPEs results, and the PCK metric computed over the HMMR results. Highest failure rates occurred in the OpenPose system mainly in the face points (eyes, ears, nose), wrists, ankles, or feet (big toe, small toe, heel) typically occluded in the player's side or back view. Unlike OpenPose, AlphaPose obtained a complete estimation of the keypoints more often, and its fails were typically caused by the inability to find the player in scenes with other people.

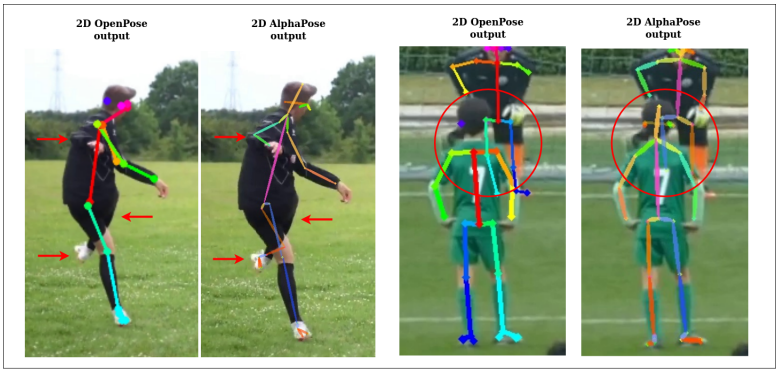


Figure 21 – Unsuccessful pose estimation examples. OpenPose (left) and AlphaPose (right) 2D pose recognition results with limb occlusion. The red arrows and circles indicate the points where the systems failed. Extracted from (LESSA; COLOMBINI; SIMÕES, 2021b).

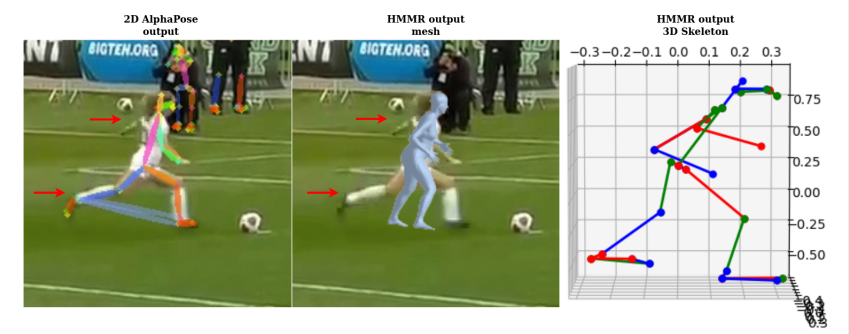


Figure 22 – Unsuccessful pose estimation examples. Incorrect 2D Alphapose recognition (right) and consequent incorrect 3D mesh recognition (center) and skeleton (right). Extracted from (LESSA; COLOMBINI; SIMÕES, 2021b).

5.2 Software Platforms

Simulation Environment

We adopted the latest stable version released of system robot simulation CoppeliaSim V4.2.0 (April 6th 2021). Although it is a commercial solution, a free student version is available. It is portable to the major operating system: Windows, Linux, and MacOS. It is also compatible with embedded scripts, plugins, nodes, Robot Operating Systems ROS interface, remote Application Programming Interface, and other solutions. As shown in Figure 24, CoppeliaSim provides several resources such as C/C++ API, customization scripts, Lua API, events callback from the simulator to plugins, remote API function, ROS transit from data exchange between CoppeliaSim and external applications, and custom connection to or from external applications (socket, serial, pipes, etc.). Also, CoppeliaSim provides four types of physical engine platforms: Bullet, Open Dynamics Engine, Vortex Studio and Newton Dynamics. Several realistic sensors are available, like proximity sensors, vision sensors and force sensors. It supports six types of programming languages (Lua, C, C++, Python, Java, Matlab, and Octave) (ROBOTICS,).

The simulated version of the humanoid robot Marta was modeled by the authors (CHENATTI et al., 2018a). Since the first model, the robot was improved with modifications in its mechanical specification for better realistic performance. Figure 25 presents the humanoid in system simulation. Marta is usually simulated using the Bullet physics

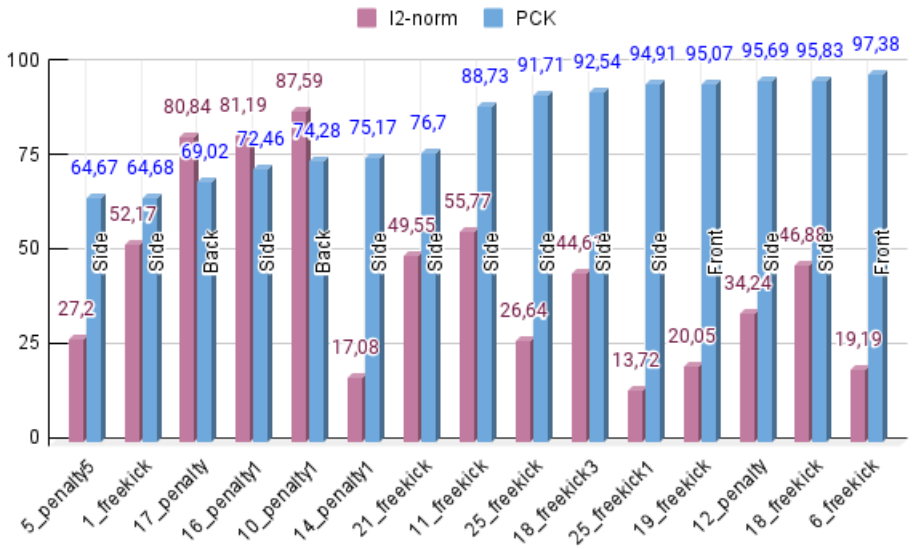


Figure 23 – Comparative results of the 15 videos that successfully passed both HMMR- AlphaPose and HMMR-OpenPose.

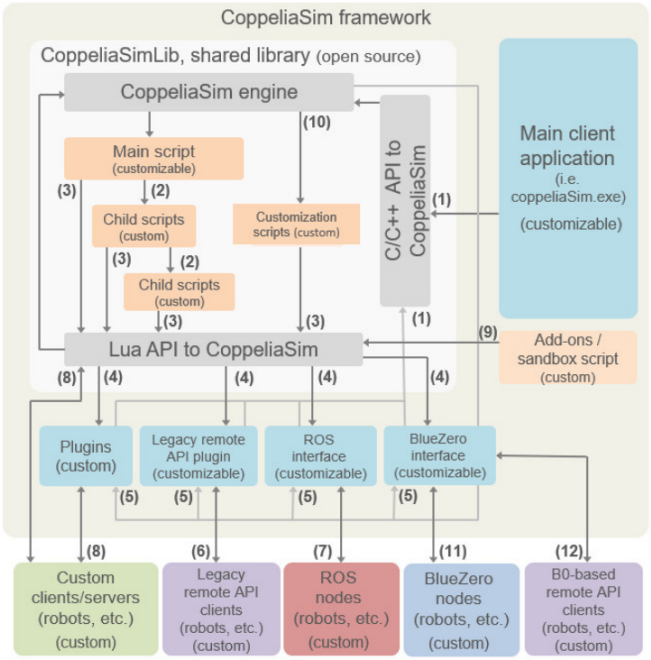


Figure 24 – Customization possibilities in CoppeliaSim control architecture (ROBOTICS,).

engine platform.

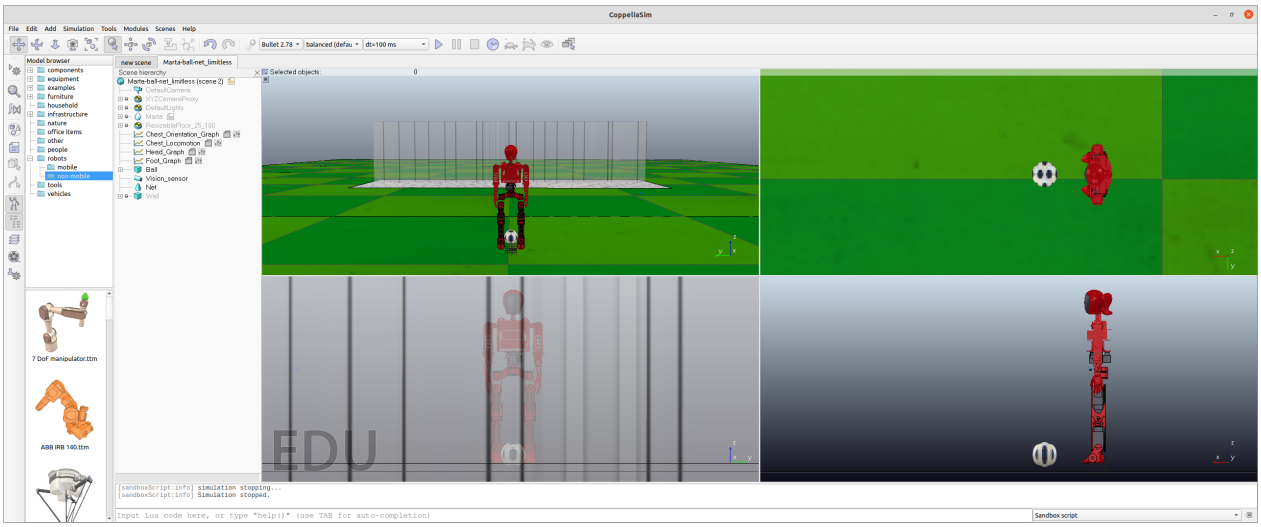


Figure 25 – Image of Marta simulated version in CoppeliaSim. From left to right we can observed the toolbar (upper), the model browse, the robot's scene hierarchy, the quadrant viewing environment and execution notifications and coding area in Lua (bottom).

Programming Languages and libraries

The present research adopts the *Python3* language at version 3.6.13. The scene of the humanoid Marta in CoopeliaSim is launched using PyRep toolkit that communicates with python-coded mimic environment, inheritance of the OpenAI Gym RL interface, and the SAC DRL algorithm is executed with the Rlkit framework. The Diagram 26 shows the main toolkits used to the RL MartaMimic environment.

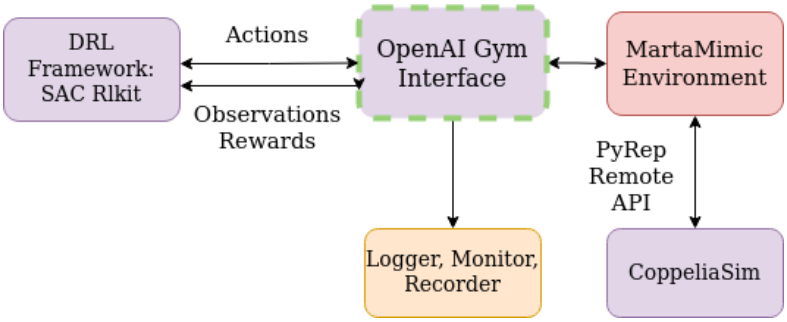


Figure 26 – MartaMimic environment and its directly related components. The OpenAI Gym Interface provides an abstraction for the environment to the Rlkit DRL SAC algorithm. The Gym interface is a mask to the MartaMimimic Environment responsible to launch the CoppeliaSim by the PyRep API and managemnt the additional libraries and RL tasks.

PyRep API

PyRep API is a toolkit developed by JAMES; FREESE; DAVISON (2019) for robot learning research. PyRep allows to launch CoppeliaSim's and execute the PyRep python code synchronously, which improves the communication speed between them. PyRep has a list of classes to arms, grippers, and mobile robots, but does not support humanoid robots

in the original approach. Given this, GONÇALVES (2021) in her research developed a class for controlling humanoid robots that could be applied to any humanoid robot, in particular the Marta humanoid robot.

Rlkit

Rlkit (BERKELEY, 2019) is a RL framework implemented in *Pytorch*, with an extensive list of implemented algorithms, such as (Double) Deep Q-Network, Temporal Difference Models, Twin Delayed Deep Determinstic Policy Gradient, Hindsight Experience Replay, Advantage Weighted Actor Critic, Soft Actor Critic. We adopt for this work the SAC algorithm provided by the Rlkit framework, which was developed within the framework of the work published by HAARNOJA et al.. 2018. The SAC algorithm (HAARNOJA et al., 2018) is presented in Algorithm 1.

Algorithm 1 Soft Actor-Critic with Automatic Entropy Adjustment

Input: θ, θ_2, ϕ ▷ Initial Parameters
$\theta_1 \leftarrow \theta_1, \theta_2 \leftarrow \theta_2$ ▷ Initialize target network weights
$D \leftarrow \emptyset$ ▷ Initialize an empty replay pool
for *each iteration* **do**
  for *each environment step* **do**
    $a_t \sim \pi_\theta(a_t|s_t)$ ▷ Sample action from the policy
    $s_{t+1} \sim p(s_{t+1}|s_t, a_t)$ ▷ Sample transition from the environment
    $D \leftarrow D \cup \{(s_t, a_t, r(s_t, a_t), s_{t+1})\}$ ▷ Store the transition in the replay pool
  for *each gradient step* **do**
    $\theta_i \leftarrow \theta_i - \lambda_Q \hat{\nabla}_{\theta_i} J_Q(\theta_i)$ for $i \in 1, 2$ ▷ Update the Q-function parameters
    $\phi \leftarrow \phi - \lambda_\pi \hat{\nabla}_\phi J_\pi(\phi)$ ▷ Update policy weights
    $\alpha \leftarrow \alpha - \lambda \hat{\nabla}_\alpha J(\alpha)$ ▷ Adjust temperature
    $\bar{\theta}_i \leftarrow \tau\theta_i + (1-\tau)\bar{\theta}_i$ for $i \in 1, 2$ ▷ Update target network weights
Output: θ, θ_2, ϕ ▷ Optimized parameters

As initially described 2.3.1, the algorithm uses two parameterized Q-functions, with parameters θ_1 and θ_2, trained them independently to minimize the Q-function loss ($J_Q(\theta_i)$):

$$\mathbb{E}_{(s_t, a_t, s_{t+1}) \sim D}\left[\left(Q_{\theta_i}(s_t, a_t) - (r(s_t, a_t) + \gamma V_{\theta_1, \theta_2}(s_{t+1}))\right)^2\right], \tag{46}$$

The value function $V_{\theta_1, \theta_2}(s_t)$ is defined using the Q-functions and the policy as $\mathbb{E}_{a_t \sim \pi_\phi}\left[\min_{i \in 1,2} Q_{\theta_i}(s_t, a_t) - \alpha log \pi_\phi(a_t|s_t)\right]$. The Gaussian Policy with parameters ϕ is learned by minimizing:

$$J_\pi(\phi) =_{s_t \sim D, a_t \sim \pi_\phi}\left[\alpha log \pi_\phi(a_t|s_t) - \min_{i \in 1,2} Q_\theta(s_t, a_t)\right]. \tag{47}$$

The dynamic temperature α, is learned by performing incomplete optimization, alternating between taking a single gradient step on each objective. The Equation 48 compute the gradients for α with following objective.

$$J(\alpha) =_{s_t \sim D, a_t \sim \pi_\phi)} [-\alpha log \pi_\phi(a_t|s_t) - \alpha]. \tag{48}$$

The Rlkit also adopt the ADAM optimization solver (PASZKE et al., 2017) for the NNs. Adam is a stochastic gradient descent that uses mini-batch optimization, which requires less memory to solve large data problems.

The explained algorithm was used for this work without modifications. The hyper-parameters used in the SAC algorithm used for our implementation are shown in Table 3.

Hyper-parameter	Value
discount factor (γ)	0.99
learning rate (α)	0.0003
replay buffer size	50000
minibatch size	256
soft update coefficient (τ)	0.005

Table 3 – SAC hyper-parameters.

OpenAI Gym

OpenAI Gym (BROCKMAN et al., 2016) is a toolkit for developing and comparing RL algorithms, compatible with any numerical computation library, that provides an abstraction for the environment which allows the implementation of different styles of the agent interface. Also, gym library contains a collection of the environment with a shared interface that allows the users to write general algorithms. We adopt the OpenAI gym toolkit to quickly set up our customized environment.

5.3 Experimental procedure

We conducted a set of experiments in order to evaluate the proposed approach in a soccer kick task. In all experiments, the simulated version of the Marta robot was positioned in an object-free environment in the CoppeliaSim simulator, and the proposed approach was applied based on a selected video that served as reference motion. In each case, a different policy and reward function was adopted. We positioned the ball about 0.3m from the right foot of the Marta robot since the reference player kicks with the right leg. In the simulated scene, the ball's weight was configured to $0.450Kg$, which is the weight normally adopted in a real-life soccer ball. We also observed that the soccer players in the videos were not positioned exactly in front of the ball in some cases. Their bodies have some rotation degree concerning the line that connects their bodies to the ball. To simplify the problem, some rotation correction was initially applied to the reference frames in some cases. Also, the position of the pelvis (body reference) was also normalized to the range 0 - 0.5m, respectively, in the first and last frames. This process is illustrated in figure 27. In some cases, we also applied a scale factor to the skeleton frames to better adjust their height to the robot height (1m).

Four experiments were carried out. In the first two experiments, we adopted two distinct approaches derived from the method Procedure -1 described in section 4.5.2 to train the two policies: π_{kick1} and π_{kick2}. The method Procedure -2, described in section 4.5.2, was adopted to train the two policies π_{kick3} and π_{kick4} in the last two experiments. These experiments are detailed next.

5.3.1 EXP-01: Imitating angles using real torque values

In this experiment, we aimed to investigate the strategy to imitate the angles of the reference frames. The experiment was configured as follows:

- General parameters:
  - Reference video: *6freekick* of the Soccerkicks dataset (25 FPS, 1.24 seconds long);
  - Frames interpolation: no;
  - Skeleton scale: no;
  - Weights (J_{w_n}): Root: 0.1; Chest: 0.1; Shoulder: 0.04; Elbow: 0.04; Upper leg: 0.2; Knee: 0.1; Goal: 0.2;

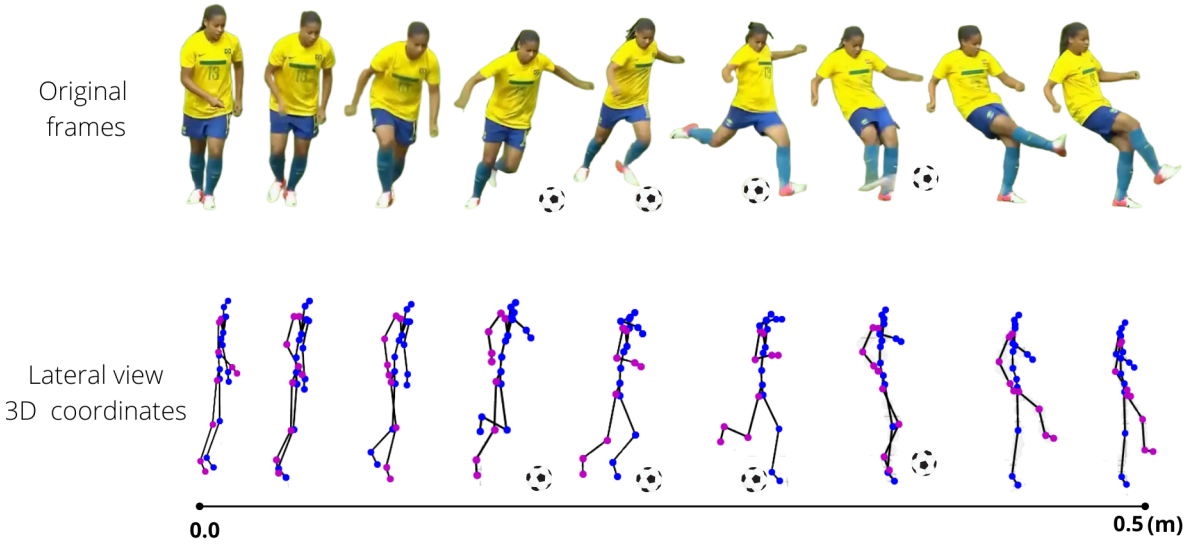


Figure 27 – Selected frames of the human soccer player Marta extracted from the *6freekick* video of SoccerKicks dataset and the respective 3D coordinates generated. The position of the pelvis was translated to be aligned to the player-ball line and the last position of the pelvis was configured to 0.5 meters in X-axis.

  - Scale factors: Pose (S_P): 2; Velocity (S_V): 0.1; End-effector (S_{EE}): 40; Root (S_{Root}): 5; Center of Mass (S_{CoM}): 10; Goal (S_G): 4;
  - Coppelia parameters: simulation step: 40ms; Maximum motors torque: **limited** to 1.5 N.m for the AX-12 (head and neck) and 7.3 N.m for the MX-64T (all other joints);
  - Time step: 40ms;

- Training parameters:
  - Training episodes: 70,300;
  - Criteria to end an episode: 1. The robot falls; 2. Episode timed out at 20s;
  - Reinforcement weights: Pose (W_{pose}): 0.5; Velocity (W_S): 0.05; End-effector (W_{EE}): 0.15; Root (W_R): 0.2; Center of Mass (W_{CoM}): 0.1; Goal (W_G): 0.3;
  - NN parameters: Number of hidden layers: 2; Neurons per layer: 512, 512, 1; Activation functions: Relu; Learning rate (α): 0.0003; Batch size: 256;
  - SAC parameters: Discount factor (γ): 0.99; Soft update coefficient (τ): 0.005;
  - State-space: proprioceptive, sensory and additional observation inputs according to section 4.6.1.1;
  - Reward (R): The total reward is composed of:
    1. Pose reward (R_{Pose}): reward related to the distance between the current and reference **angles** of the target joints according to equations 25, 36 and 37. J_{wn} weights were set to zero for the following joints: Leg Yaw, Leg Roll, Neck and Head;
    2. Root reward (R_{Root}): reward related to the rotation, position, angular and linear velocities of the robot pelvis when compared to the reference frame according to equations 26, 31, 32, 33, 34 and 35;

3. Target-joints velocity reward (R_{Vel}): reward related to the velocity of the target-joints when compared to the reference frame according to equations 27, 38 and 39;
4. End-effector reward (R_{EE}): reward related to the ankle position when compared to the reference frame according to equations 28, 40 and 41;
5. Center of Mass reward (R_{CoM}): reward related to the velocity of the robot center of mass when compared to the reference frame according to equations 29, 42 and 43;
6. Goal reward (R_G): reward related to the distance between the initial position of the ball and the robot reference foot according to equations 30, 44 and 45.

– RL Learned Policy: π_{kick1};

5.3.2 EXP-02: Imitating angles using unlimited torque values

In the second experiment, we aimed to investigate the impact of the motor torque limitations on the robot movements. We replicated the previous simulation but with unlimited torque value in Marta's joints (maximum values allowed by CoppeliaSim). The experiment was configured identically to the first, except:

- General parameters:
  - Coppelia parameters: simulation step: 40ms; Maximum motors torque: **unlimited** for all motors;
  - Time step: 40ms;
- Training parameters:
  - Training episodes: 66,160;
  - RL Learned Policy: π_{kick2};

5.3.3 EXP-03: Imitating joints position, unlimited torque and some joints in Pose Reward

In the third experiment, we hypothesized that since Marta's body scheme is significantly different from the human reference body scheme, the angles could not be the best choice to compare robot and reference poses. In this way, the experiment was configured identically as the previous, except:

- General parameters:
  - Frames interpolation: yes, according to section 4.5.2;
  - Skeleton scale: yes;
  - Coppelia parameters: simulation step: 100ms; Maximum motors torque: **unlimited** for all motors;
  - Time step: 100ms;
- Training parameters:
  - Training episodes: 328,080;
    1. Pose reward (R_{Pose}): reward related to the distance between the current and reference **positions** of the target joints according to equations 25, 36 and 37; J_{wn} weights were set to zero for the following joints: Neck, Head, chest and Root (Leg Yaw and Roll were used in the reinforcement);
  - RL Learned Policy: π_{kick3};

5.3.4 EXP-04: Imitating joints position, unlimited torque, and all joints in Pose Reward

In the fourth experiment, we investigated the robot's performance using all joints in the Pose Reward (P_R). This experiment was configured identically to the previous, except:

- General parameters:
  - Frames interpolation: yes, according to section 4.5.2;
  - Skeleton scale: yes;
  - Coppelia parameters: simulation step: 50ms; Maximum motors torque: **unlimited** for all motors;
  - Time step: 50ms;
- Training parameters:
  - Training episodes: 92,600;
    1. Pose reward (R_{Pose}): reward related to the distance between the current and reference **positions** of the target joints according to equations 25, 36 and 37; J_{wn} weights were set to zero for the following joints: Neck, Head(Leg Yaw, Leg Roll, Chest and Root were used in the reinforcement);
  - RL Learned Policy: π_{kick4};

6 RESULTS AND DISCUSSION

This chapter presents and discusses the experimental results of each experiment detailed in previous chapters. After the training, we generated graphs with 20 roll-outs of each policy to compare the results to analyze the trajectories from the pelvis, ankles, and CoM. To analyze the measurements of chest stability and distance to the ball, we generated graphs with 50 roll-outs of each policy. Screenshots were generated with the time lapses plotted using distinct colored lines. The results are shown in the next sections.

6.1 EXP-01: Imitating angles using real torque values

Figure 28 presents the total reward obtained in each episode during the training process with a moving average of 100 episodes. The maximum reward obtained during the training was $1.04x10^3$. The total average reward achieved during the policy training was $67.7x10^1$. Figure 29 details the contribution of each component of the reward function. Figure 30 presents a better view of each of the components of the reward function.

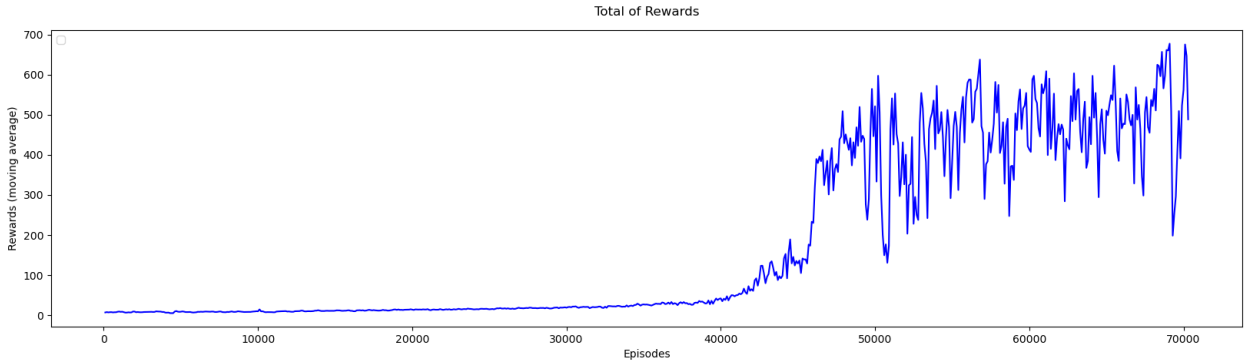


Figure 28 – Average total Reward (R) obtained at each 100 episodes during the training of EXP-01.

We can observe that the Goal reward (R_G) (run towards the ball and kick it), followed by the Pose imitation reward (R_P) are the main components of the total reward (R). The CoM reward (R_{CoM}), Root reward (R_R) and End-effector reward (R_{EE}) have a minor participation in the total reward. The robot Marta was primarily encouraged to reach the ball and secondarily to imitate the reference pose. We attempt to balance both values so that one is not achieved to the detriment of the other. We can also observe a tendency that the root reward decreases in priority as the time passes. Finally, the pose velocity reward function has minor participation in the training. This means that Marta is not improving its velocity. For example, this can indicate several situations, such as it has reached a point of stability or the episode's duration is too long.

Figure 31 presents each joint contribution for the Pose reward (R_P). We can observe that the chest, elbows, and pelvis had the most relevant participation. Hence, the reference pose for these members was reached first. Next, the pose for the left shoulder and both knees was reached, followed by both legs. This configuration is because the upper joints and knees poses are more easily reached since their movements have a minor complexity. On the other hand, the legs movements, particularly the right leg, are the most difficult to train. This typically occurs because only the Pitch is being imitated from the reference frames. At the same time, all other joints must learn by trial and error.

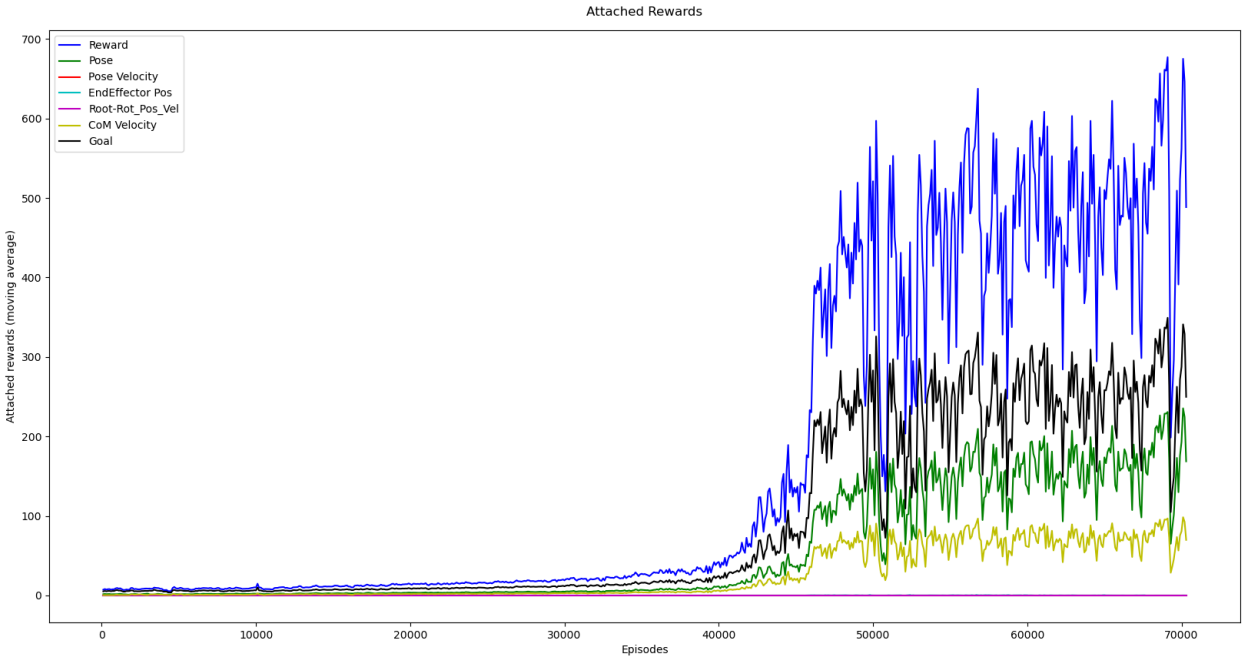


Figure 29 – Contribution of each type of reward to the total reward (R) obtained per episode during the training of EXP-01. Reward axes are shown using a moving average of size 100.

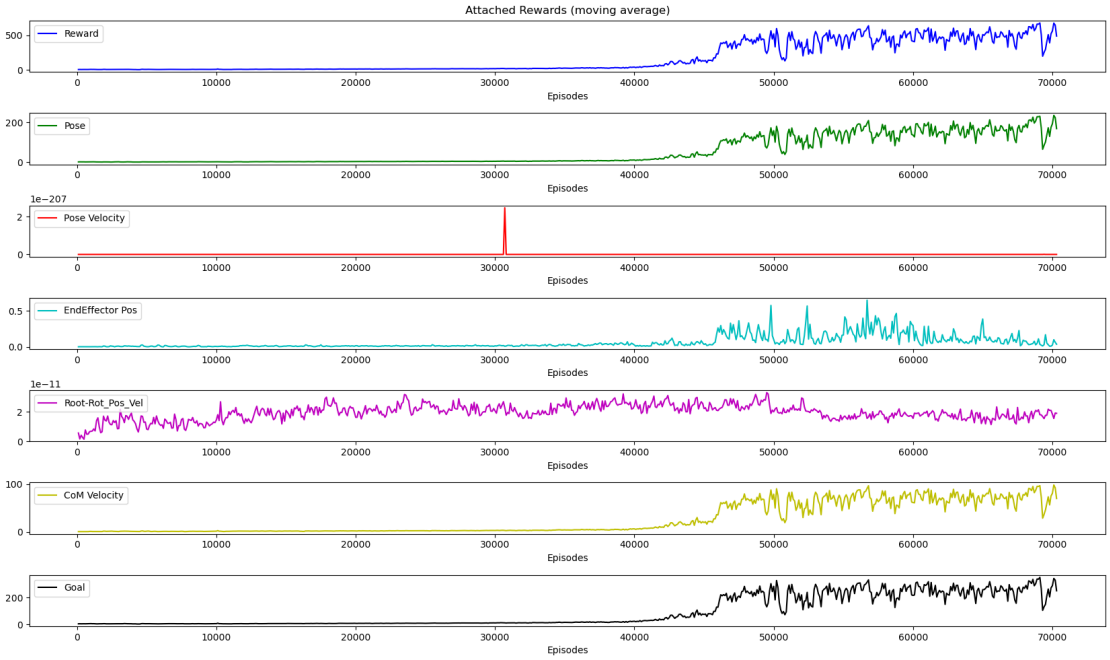


Figure 30 – Detail of the total Reward (*R*) subcomponents obtained per episode during the EXP-01. Reward axes are shown using a moving average of size 100.

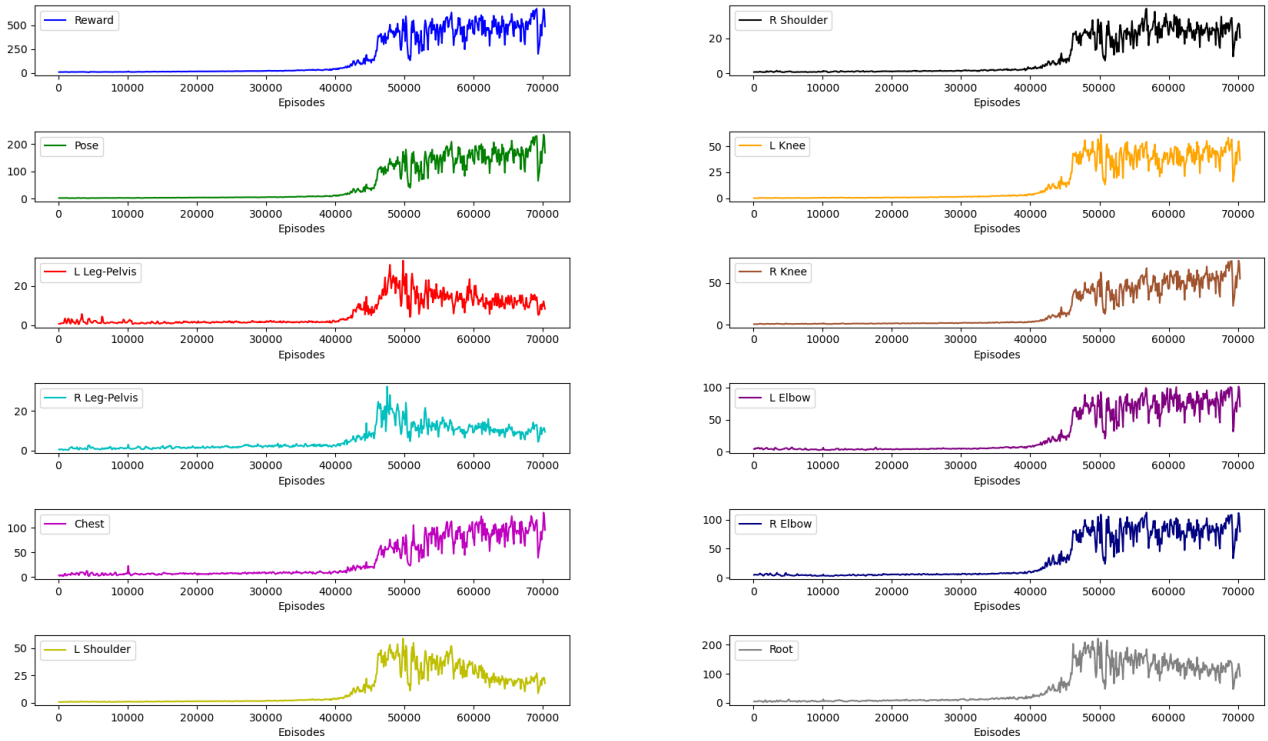


Figure 31 – Contributions of each joint to the average Pose Reward (R_P) obtained per episode during EXP-01. Reward axes are shown using a moving average of size 100.

After the training phase, we performed several tests with the learned policy π_{kick1}. The time-lapse of the motion imitation policy can be seen in Figure 32. Although Marta could not touch the ball in this experiment, we can observe that the posture is close to the reference, highlighting the chest, root, arms, and knees positions.

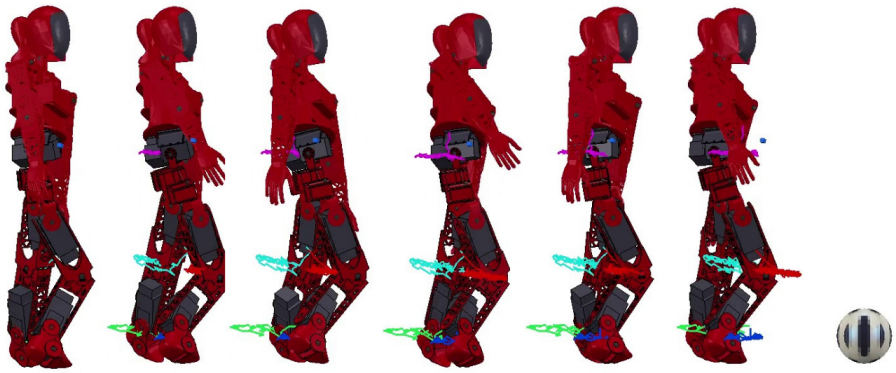


Figure 32 – Time-lapse of the motion imitation for the policy π_{kicko}. Colors indicate the trajectories of the pelvis (purple), right ankle (red), right foot (blue), left ankle (light green) and left foot (green).

Figure 33 details the trajectory of the right ankle position of the Marta robot for 20 runs with the learning policy π_{kick1}. The red line shows the reference trajectory for the right ankle. Since the reference, in this case, was not rescaled to the robot height, Marta's leg is clearly unable to reach the same trajectory. Also, considering that we are working in this training only with the leg pitch joints, the trajectory should be adapted anyway.

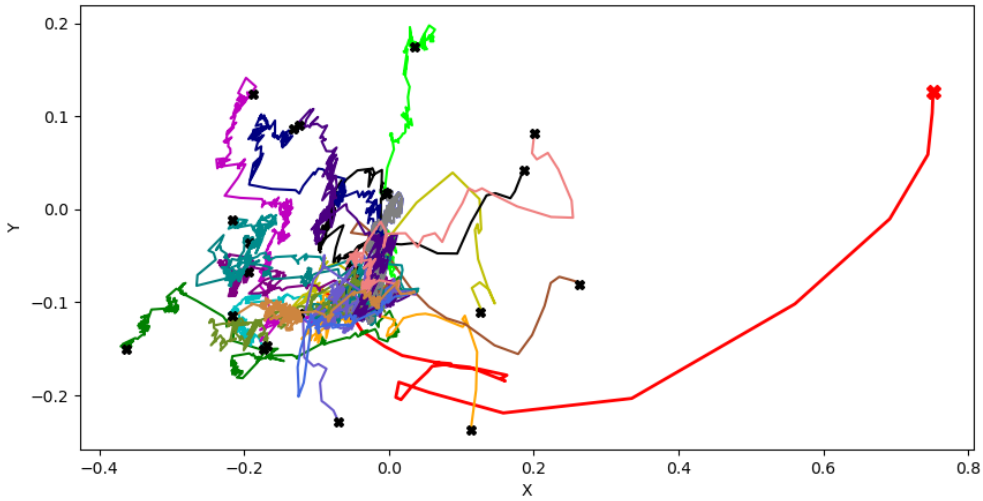


Figure 33 – X-Y (top) view of the trajectory of the right (kicking) Leg Ankle Pitch for 20 runs of policy π_{kick1} (in meters). The Read line is represents the reference trajectory.

Figure 34 presents the robot's pelvis trajectory compared to the reference (in red) for 20 runs with the policy π_{kick1}. Here, we can see that Marta was stimulated to travel approximately the same distance as the reference.

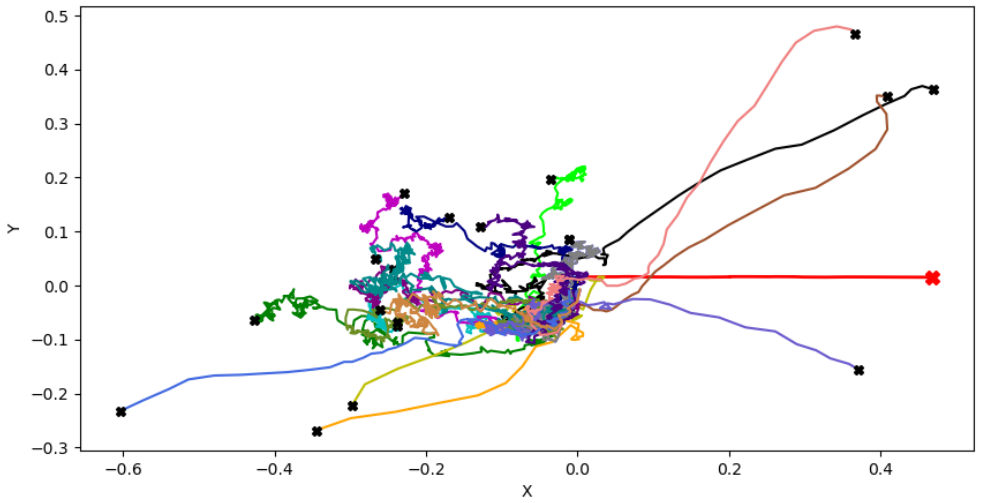


Figure 34 – X-Y (top) view of the trajectory of the root (Hip Yaw) for 20 runs with the learned policy π_{kick1} (in meters). The red line represents the reference trajectory.

The analysis of the chest Z-Axis position in Figure 35 shows that Marta's torso reached an upright posture a significant number of times with the learned policy π_{kick1}, falling a few times during the 50 runs. This result means that Marta could stabilize the waist joints many times using the reference frames, which is a great achievement for a bipedal robot due to the difficulty in balancing the three revolute joints.

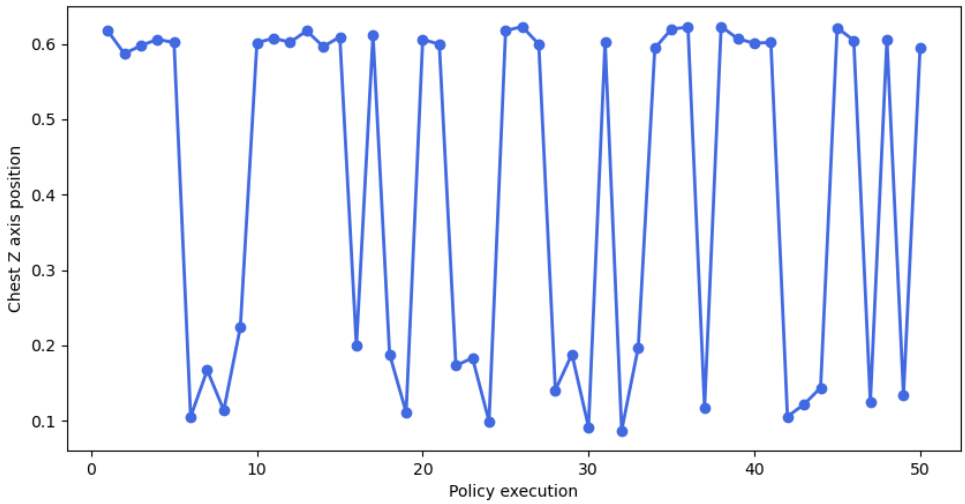


Figure 35 – Final position of the chest z-axis in meters during 50 runs with the learned policy π_{kick1}. The robot was able to stay upright during 58% of the tests.

Finally, figure 36 shows that Marta did not get close to the ball in most of the 50 executions of the learned policy π_{kick1}. In this way, although Marta reached relatively good postures imitating the reference frames, it failed to kick the ball.

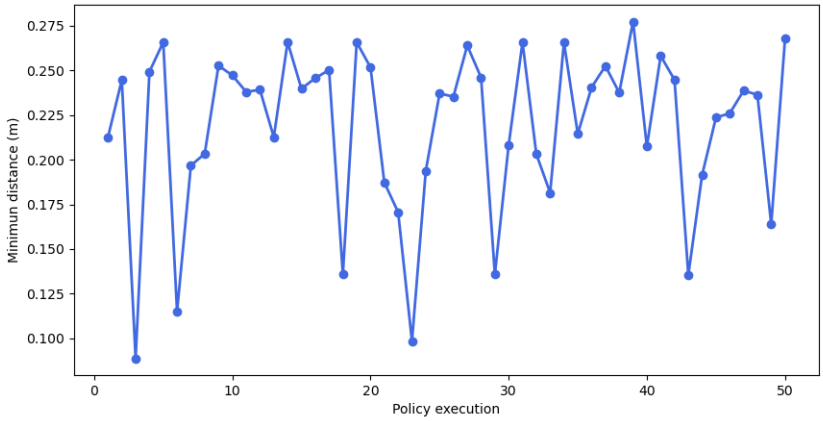


Figure 36 – Minimum distance between the right (kick) foot and the ball position in meters for 50 runs with the learned policy π_{kick1}.

6.2 EXP-02: Imitating angles with unlimited torque values

Figure 46 shows the total Reward (R) obtained per episode during the training process. The maximum total reward achieved during the policy training was $1.02x10^3$, and the maximum average reward value was 700. We can observe that the learning convergence occurs relatively later when compared to EXP-01. Since, in the current experiment, Marta had more capacity (torque) to control its body, we can assume that the robot learned a more complex policy.

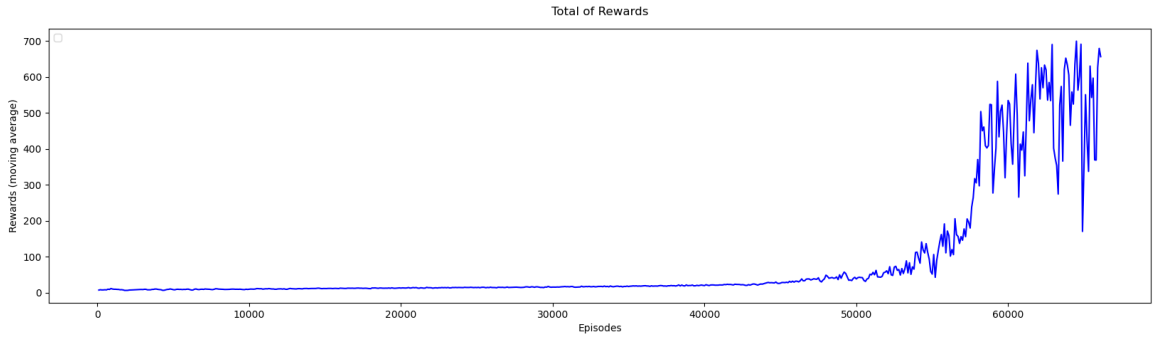


Figure 37 – Average total Reward (R) obtained at each 100 episodes during the training of EXP-02.

As shown in Figure 38 and Figure 39, one more time, Marta attempted to reach the ball while trying to reproduce the movements of the reference frames in a learning dynamics close to the EXP-01. The view of each joint component of the Pose reward function is presented in Figure 40.

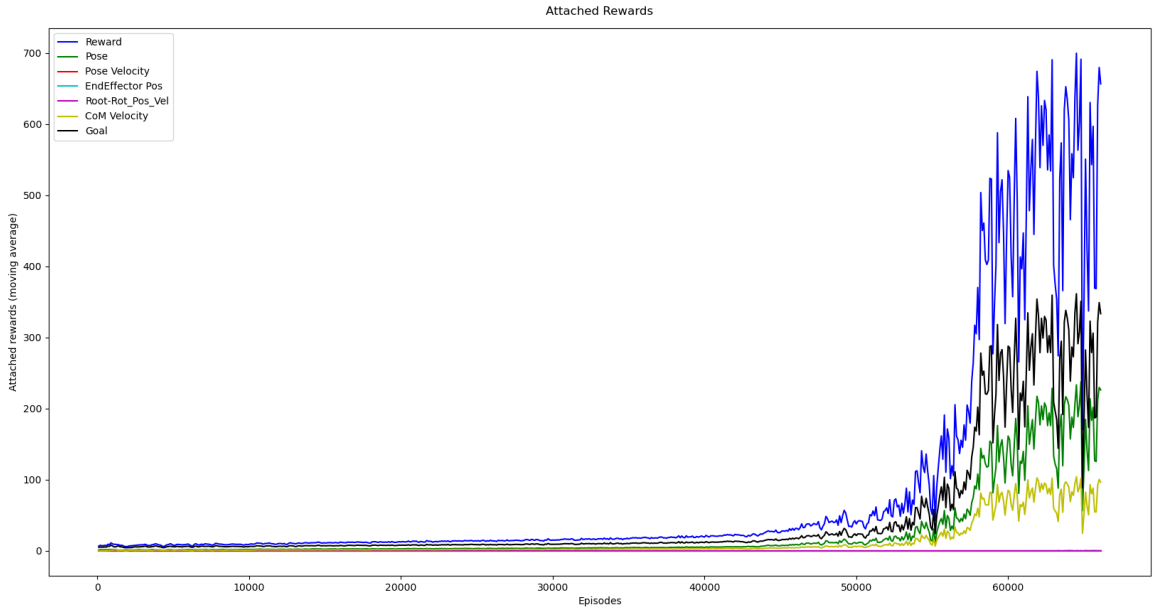


Figure 38 – Contribution of each type of reward to the total Reward (R) obtained per episode during the training of EXP-02. Reward axes are shown using a moving average of size 100.

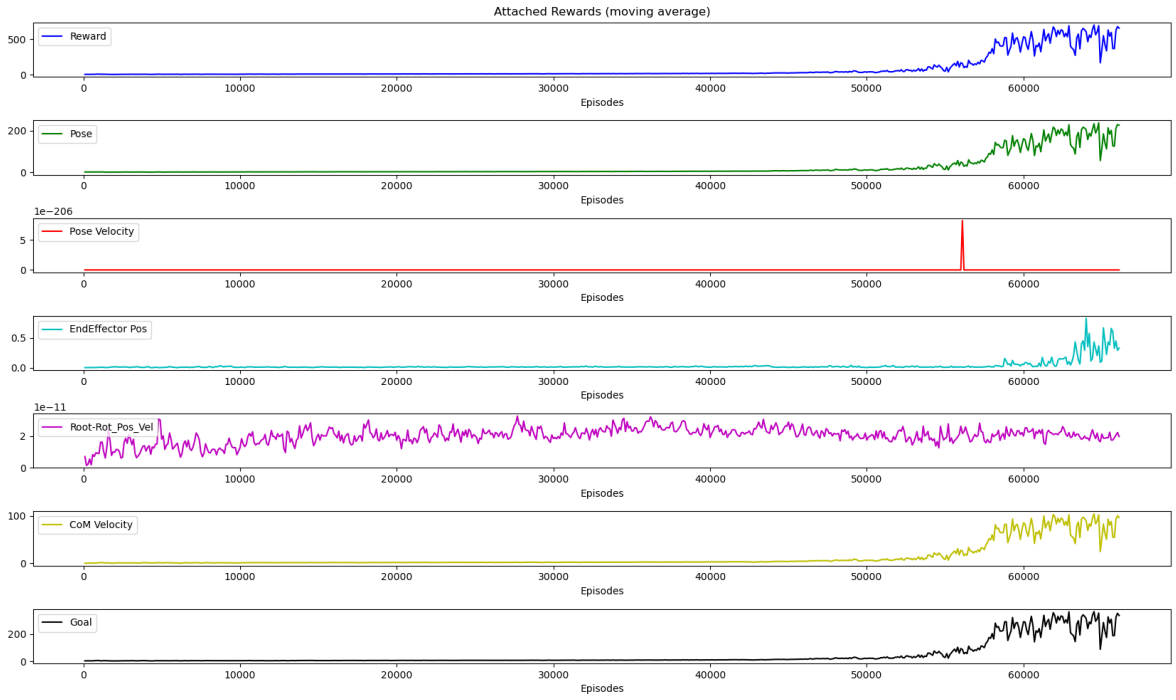


Figure 39 – Detail of the total Reward (R) subcomponents obtained per episode during the EXP-02. Reward axes are shown using a moving average of size 100.

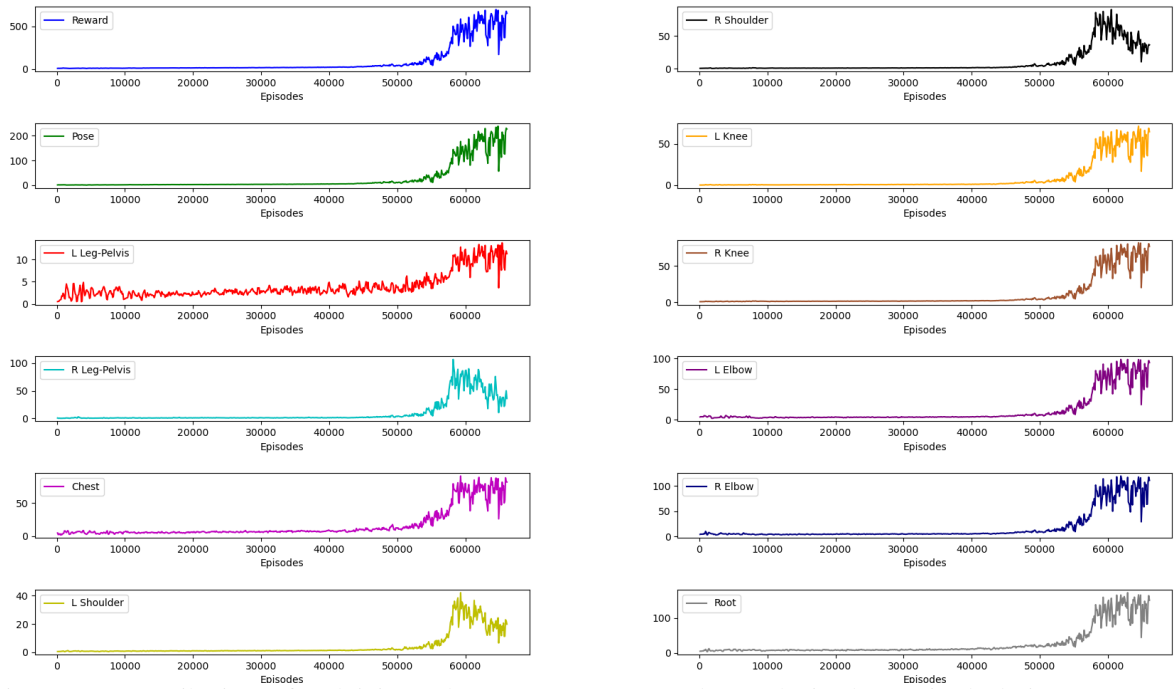


Figure 40 – Contributions of each joint to the average Pose Reward (R_P) obtained per episode during EXP-02. Reward axes are shown using a moving average of size 100.

We can observe that differently of EXP-01, the right leg, both elbows, and pelvis (root) are the most relevant rewards in R. The right leg reward is the largest since it is the most relevant member for the reference frames. Next, the chest, shoulders, and knee came right behind, followed by the left leg. We can assume that, given the unlimited torque power available, Marta can control the right leg more easily and, consequently, the foot, which led her to achieve greater rewards due to the greater focus on this limb to reach the ball. Once finished the training phase, figure 41 shows that using the learned policy π_{kick2}, Marta is on the way to achieving her goal of combining posture and hitting the ball. Figure 42 and Figure 43 presents, respectively, the right ankle and the hip trajectories. Figure 44 shows that Marta was able to keep her posture about 46% of the 50 runs using the learned policy π_{kick2}. Despite that, Marta's foot came closer to the ball, as we can see in Figure 45.

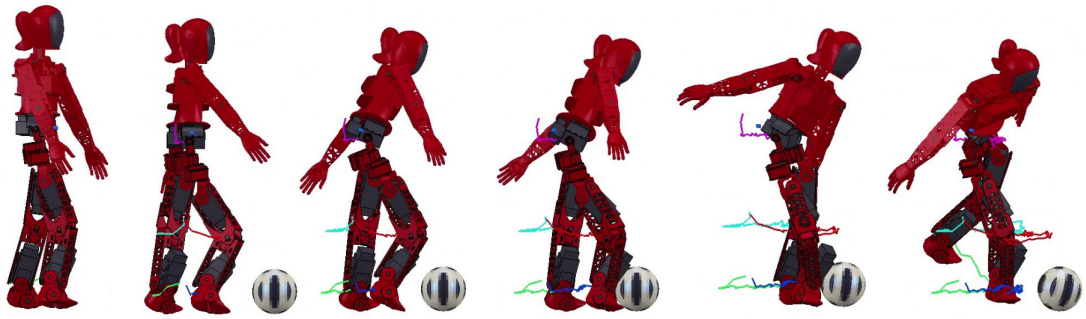


Figure 41 – Time-lapse of the motion imitation for the policy π_{kick2}. Colors indicate the trajectories of the pelvis (purple), right ankle (red), right foot (blue), left ankle (light green) and left foot (green).

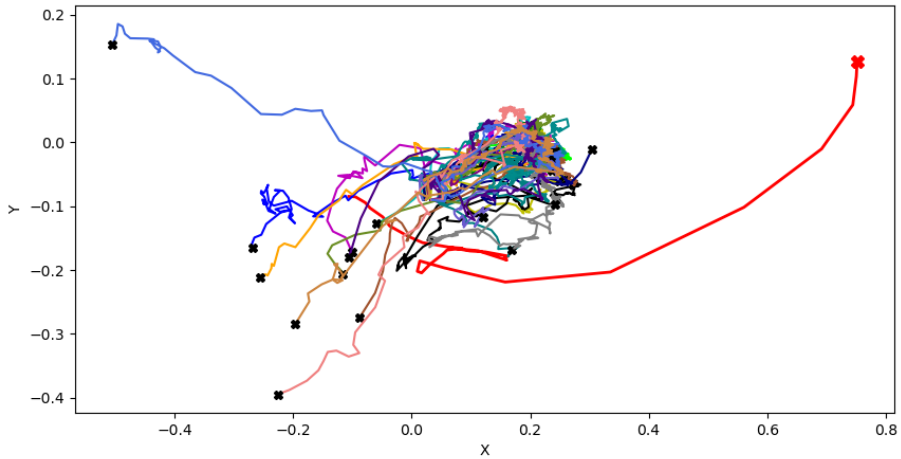


Figure 42 – X-Y (top) view of the trajectory of the right (kicking) Leg Ankle Pitch for 20 runs of policy π_{kick2} (in meters). The Read line is represents the reference trajectory.

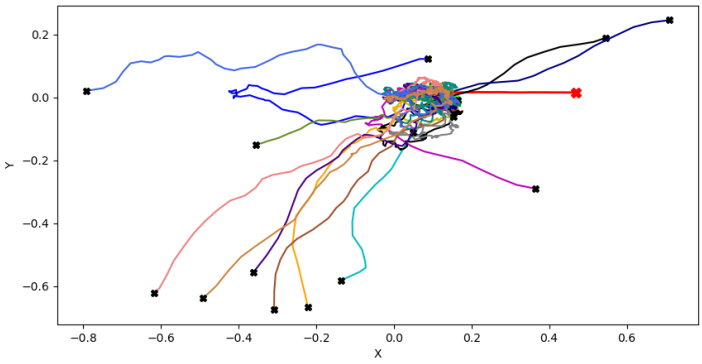


Figure 43 – X-Y (top) view of the trajectory of the root (Hip Yaw) for 20 runs with the learned policy π_{kick2} (in meters). The red line represents the reference trajectory.

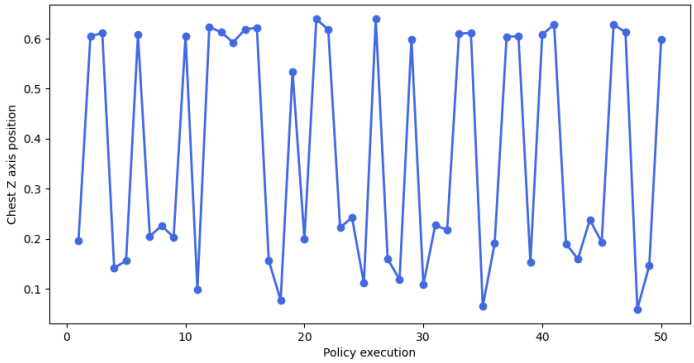


Figure 44 – Final position of the chest z-axis in meters during 50 runs with the learned policy π_{kick2}. The robot was able to stay upright during 43% of the tests.

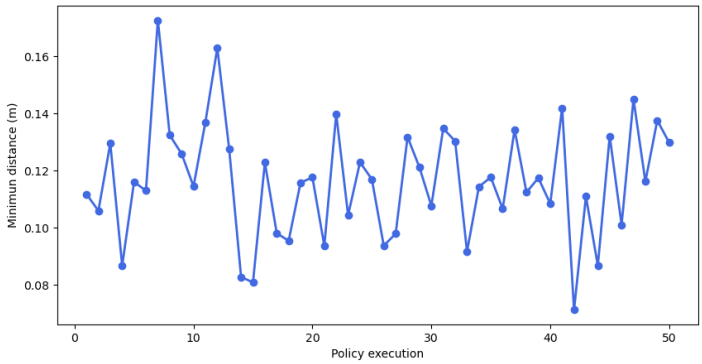


Figure 45 – Minimum distance between the right (kick) foot and the ball position in meters for 50 runs with the learned policy π_{kick2}.

6.3 EXP-03: Imitating joints position, unlimited torque and some joints in Pose Reward

Figure 46 shows the total moving average Reward (R) obtained per episode during the training. The training lasted $328x10^3$ episodes. The maximum total reward achieved during the policy training was $1.23x10^3$, with a total average reward of 65.72.

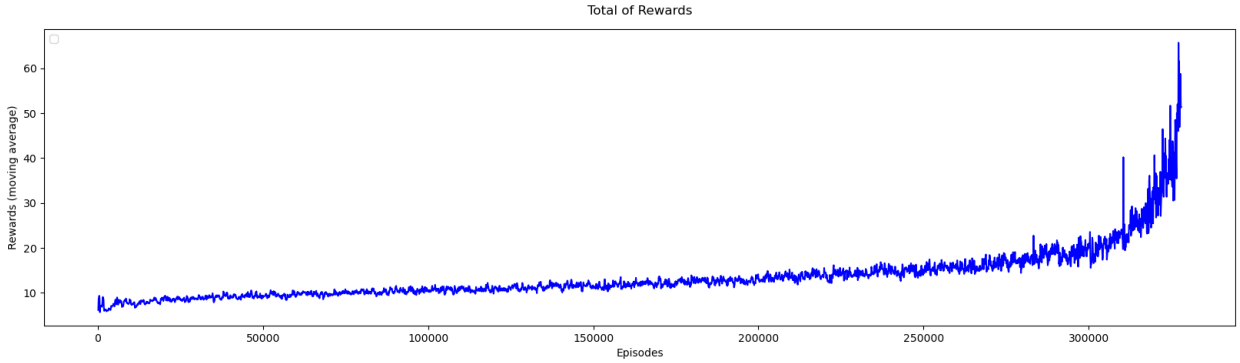


Figure 46 – Average total Reward (R) obtained at each 100 episodes during the training of EXP-03.

Figure 47 and Figure 48 shows that the Pose reward (R_P) had more importance for the training process than the Goal reward (R_G), followed by Root reward (R_R), CoM reward (R_{CoM}), target-joints velocity reward (R_{vel}) and End-effector reward (R_{EE}), respectively. Figure 49 presents the contribution of each joint to the Pose Reward (P_R). In this scenario, all joint positions increased with approximately the same importance.

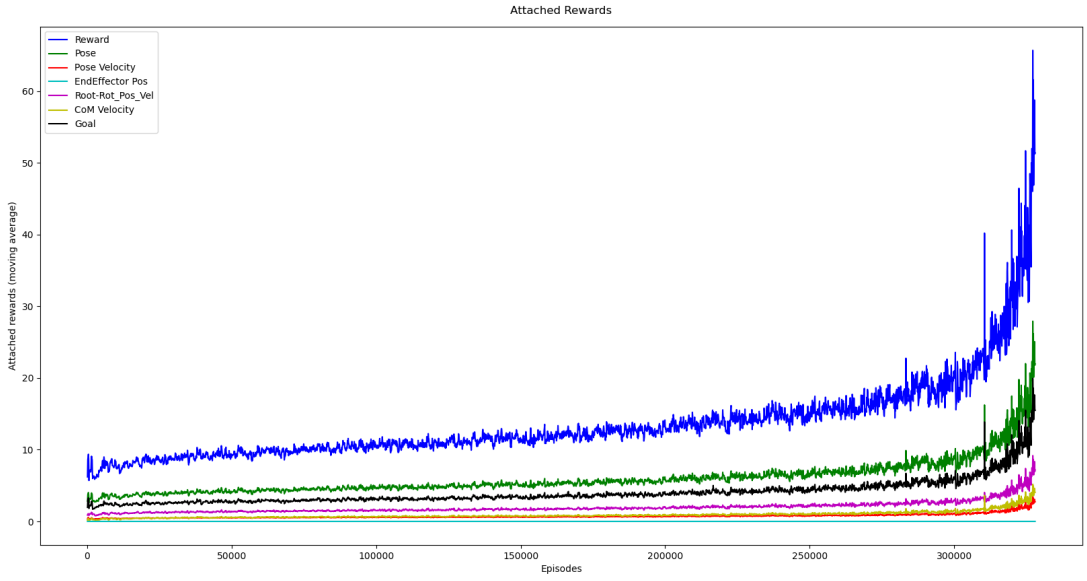


Figure 47 – Contribution of each type of reward to the total Reward (R) obtained per episode during the training of EXP-03. Reward axes are shown using a moving average of size 100.

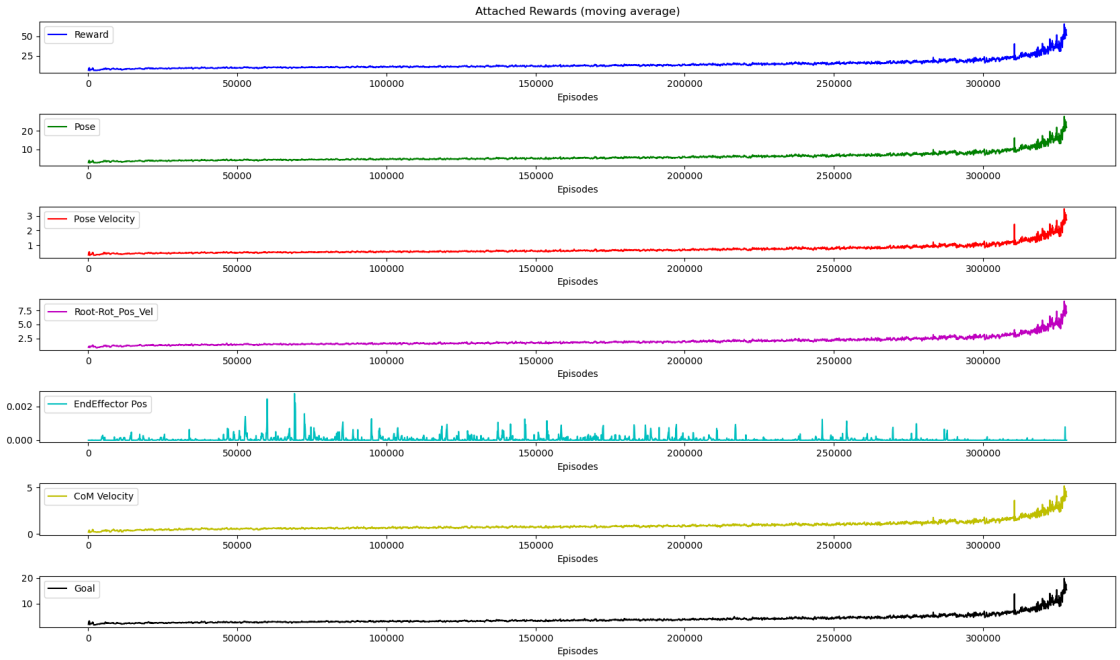


Figure 48 – Detail of the total Reward (R) subcomponents obtained per episode during the EXP-03. Reward axes are shown using a moving average of size 100.

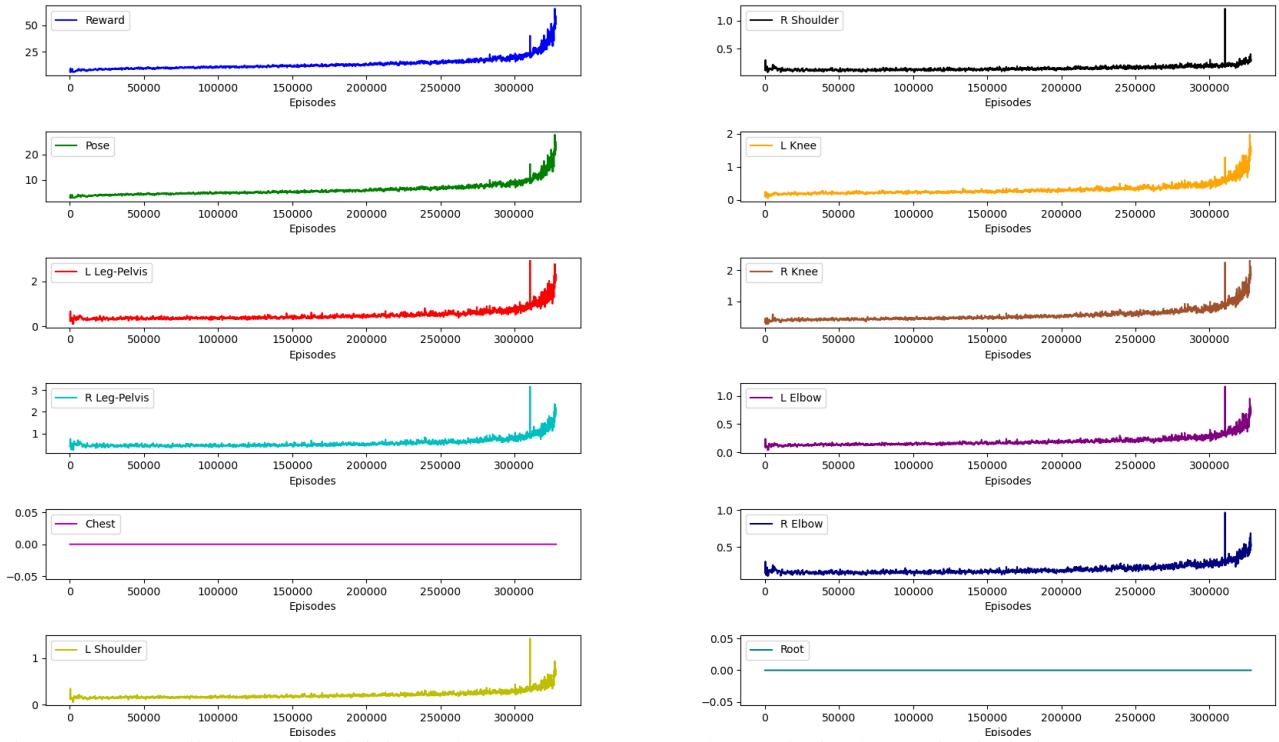


Figure 49 – Contributions of each joint to the average Pose Reward (R_P) obtained per episode during EXP-03. Reward axes are shown using a moving average of size 100.

After the training phase, we conducted some investigation with the learned policy π_{kick3}. Marta finally hit the ball with movements that resemble the reference frames in some cases. However, the upright position was not fully maintained during the movement. This was probably caused by the lack of chest and root reinforces. Also, results have shown that using the Roll and Yaw joints in the Leg allowed for a movement more similar to the reference for the right (kick) leg. This approach seems more promising than the previous approaches.

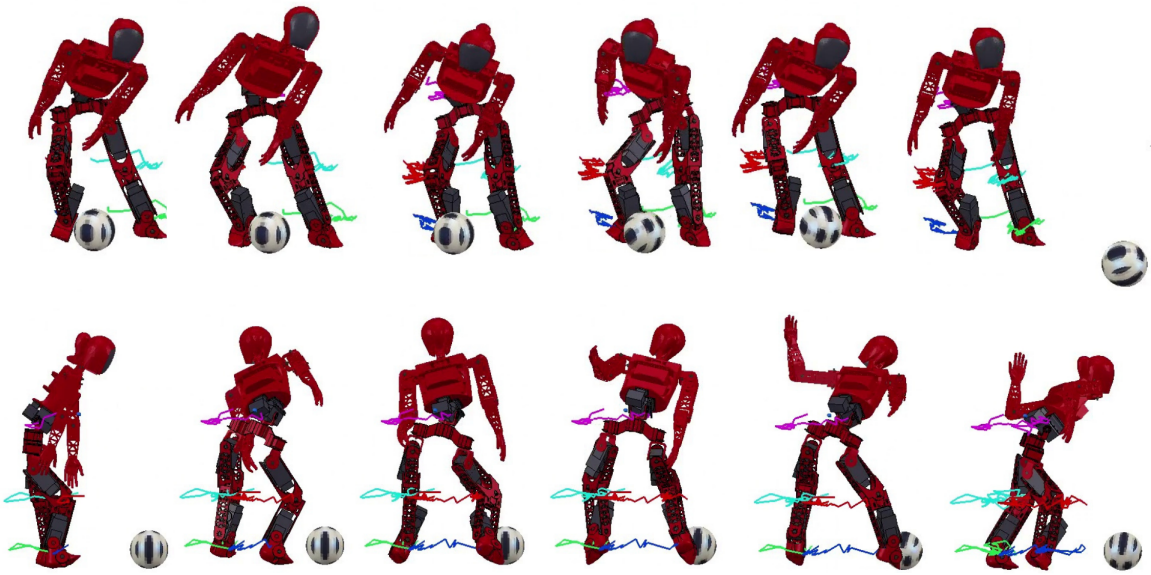


Figure 50 – Time-lapse of the motion imitation for the policy π_{kick3}. Colors indicate the trajectories of the pelvis (purple), right ankle (red), right foot (blue), left ankle (light green) and left foot (green).

As expected, since we rescaled the reference coordinates to the robot height, Marta got close to the same distance of the ankle joints reference trajectory, as shown in Figure 51.

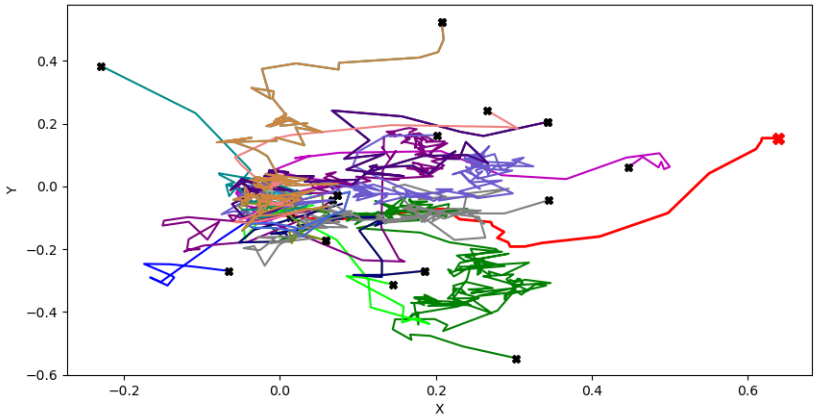


Figure 51 – X-Y view of the trajectory of the right (kicking) Leg Ankle Pitch for 20 runs of policy π_{kick3} (in meters). The Read line is represents the reference trajectory.

Also, the pelvis reference trajectory was reached a couple of times in 20 runs with the learned policy π_{kick3}, as shown in Figure 52. Marta's attempts lead to an unstable posture, causing her to fall many times, as shown in the chest Z-axis presented in Figure 53.

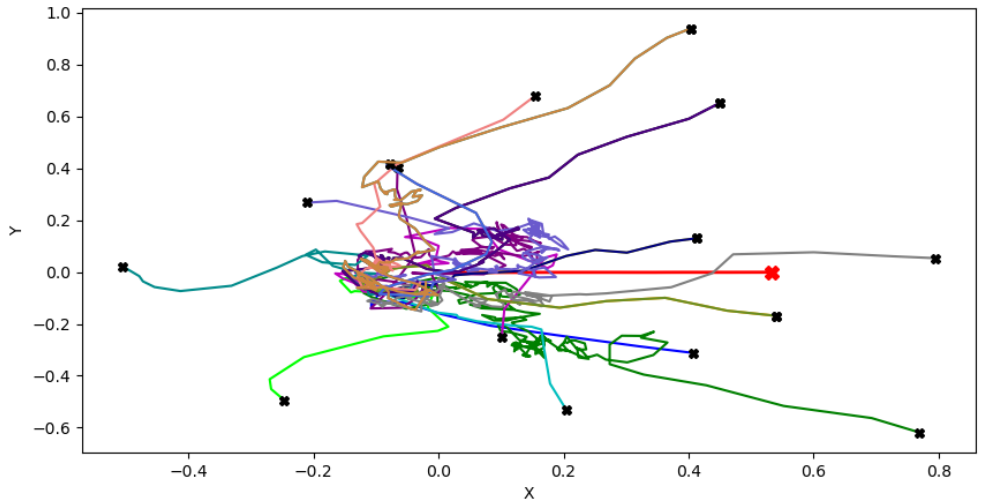


Figure 52 – X-Y view of the trajectory of the root (Hip Yaw) for 20 runs with the learned policy π_{kick3} (in meters). The red line represents the reference trajectory.

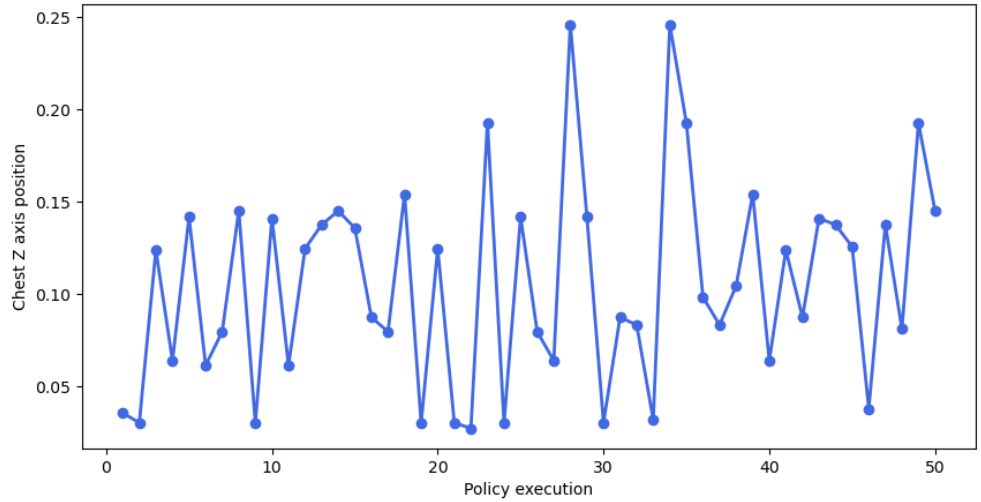


Figure 53 – Final position of the chest z-axis in meters during 50 runs with the learned policy π_{kick3}. The robot was able to stay upright during only 4% of the tests.

In Figure 54 we notice that the ball is reached 14 times in 50 runs (28%) using the learned policy π_{kick3}. The maximum distance covered by the ball after the kick was 1.4 meters on the X-axis, and 1.6 meters on the Y axis, as shown in Figure 55.

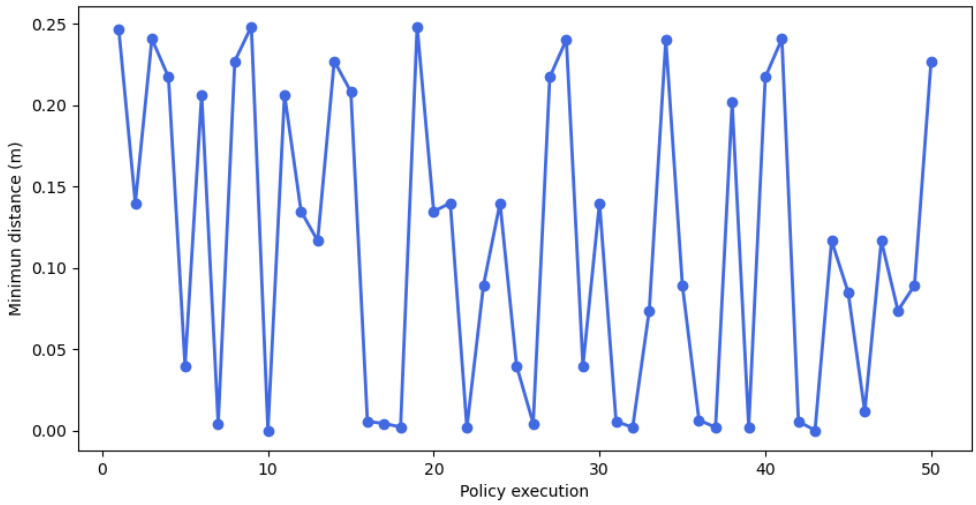


Figure 54 – Minimum distance between the right (kick) foot and the ball position in meters for 50 runs with the learned policy π_{kick3}.

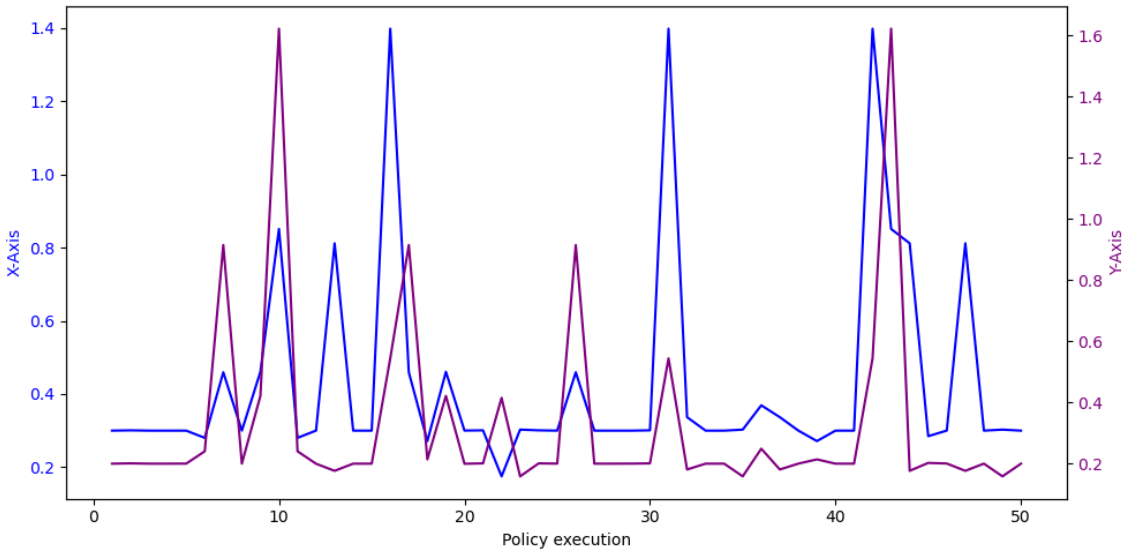


Figure 55 – Distance traveled by the ball (in meters) for 50 runs using the learned policy π_{kick3}.

6.4 EXP-04: Imitating joints position, unlimited torque values, and all joints in Pose Reward

Figure 56 presents the results for the EXP-04. The maximum reward reached during the training was 290.60, and the total moving average reward obtained per episode during the training process was 113.24. Differently from π_{kick3}, the training was obtained in fewer training steps, probably due to the adoption of the default time step that leads to a more stable training. Figure 57 and Figure 57 once again show a balance in rewards between imitating the pose and reaching the goal. Differently from the π_{kick3}, the chest position was recorded. Results presented in Figure 60 demonstrate that Marta reached a better posture of the upper body with the chest incentive in the reward function to imitate the reference.

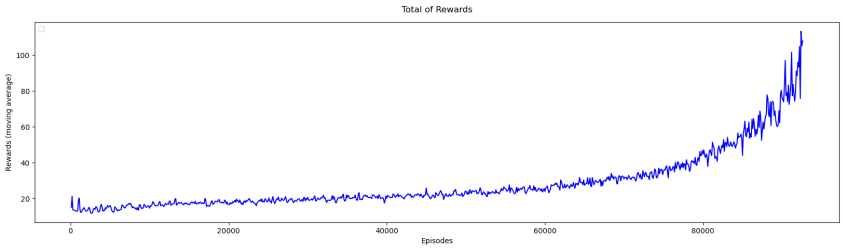


Figure 56 – Average total Reward (R) obtained at each 100 episodes during the training of EXP-04.

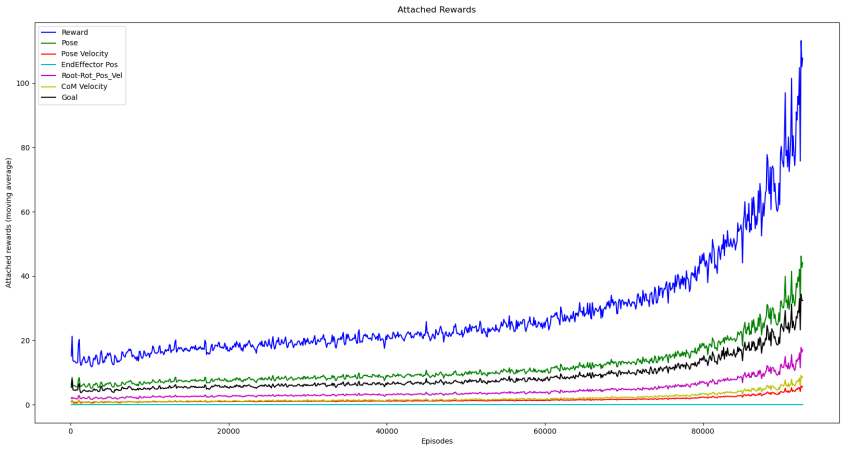


Figure 57 – Contribution of each type of reward to the total Reward (R) obtained per episode during the training of EXP-04. Reward axes are shown using a moving average of size 100.

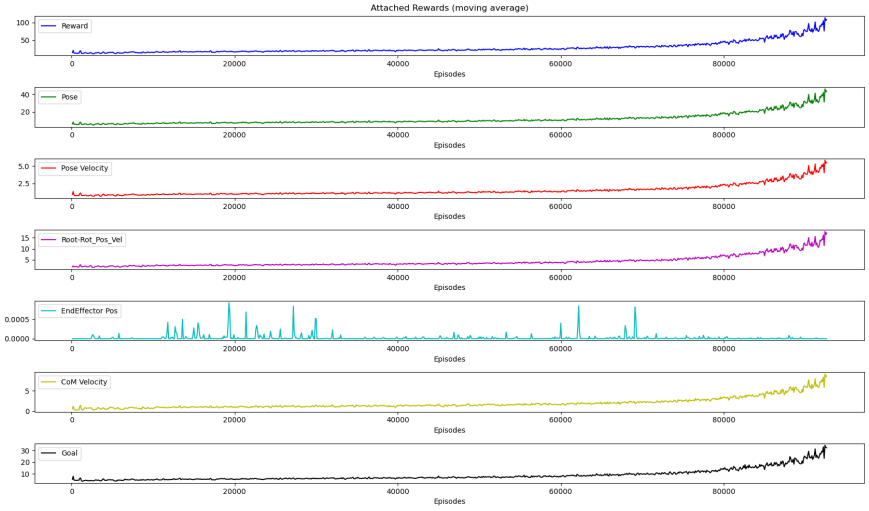


Figure 58 – Detail of the total Reward (R) subcomponents obtained per episode during the EXP-04. Reward axes are shown using a moving average of size 100.

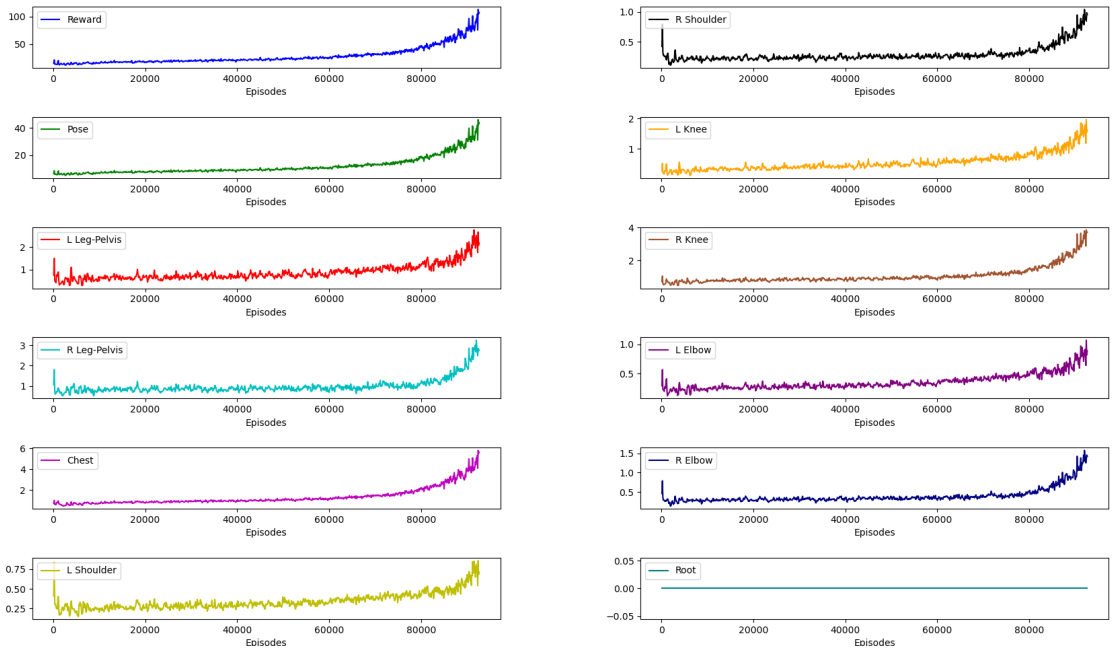


Figure 59 – Contributions of each joint to the average Pose Reward (RP) obtained per episode during EXP-04. Reward axes are shown using a moving average of size 100.

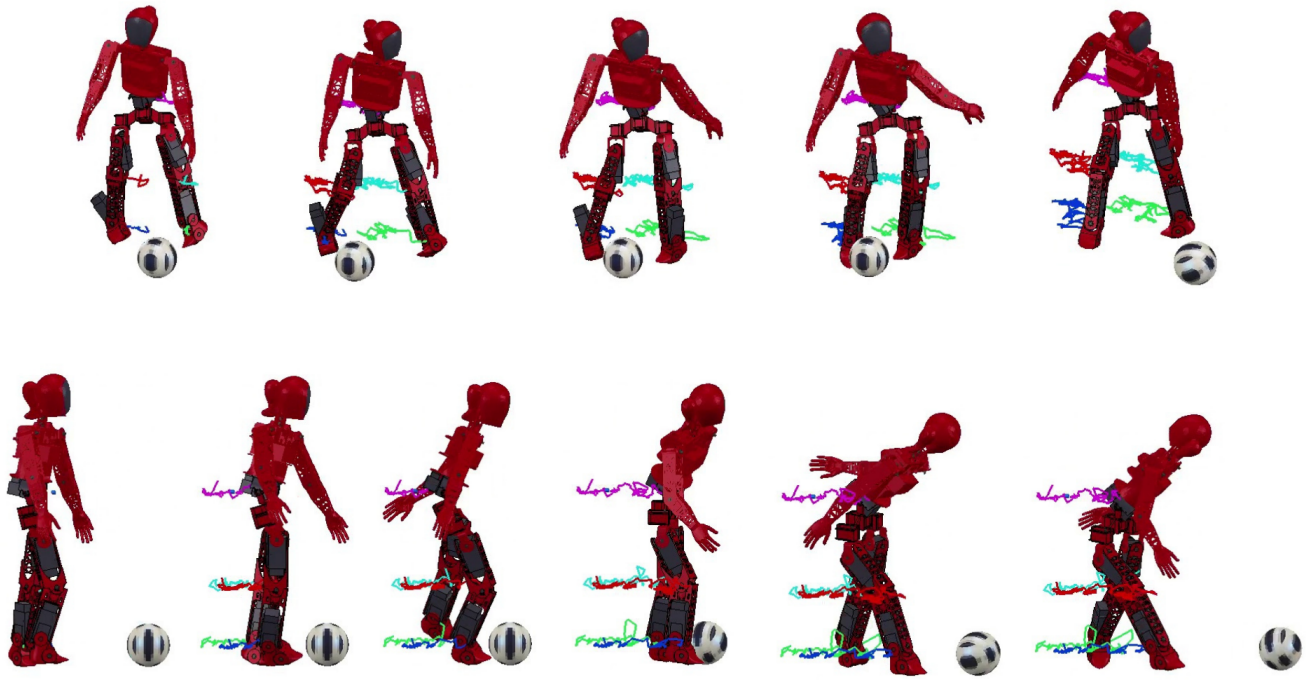


Figure 60 – Time-lapse of the motion imitation for the policy kick3. Colors indicate the trajectories of the pelvis (purple), right ankle (red), right foot (blue), left ankle (light green) and left foot (green).

As shown in the graphs 61 and 62, Marta had an improvement in following the reference trajectory with the ankle. However, given the differences in the body shape, Marta is not expected to implement exactly the same trajectory. For the pelvis, Marta has often reached the same or higher distance when compared to the reference Pelvis position.

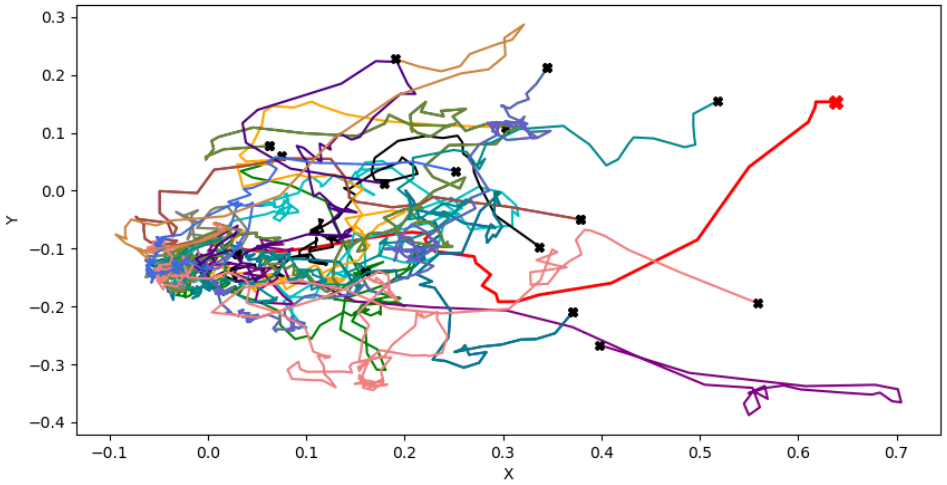


Figure 61 – X-Y view of the trajectory of the right (kicking) Leg Ankle Pitch for 20 runs of policy π_{kick4} (in meters). The Read line is represents the reference trajectory.

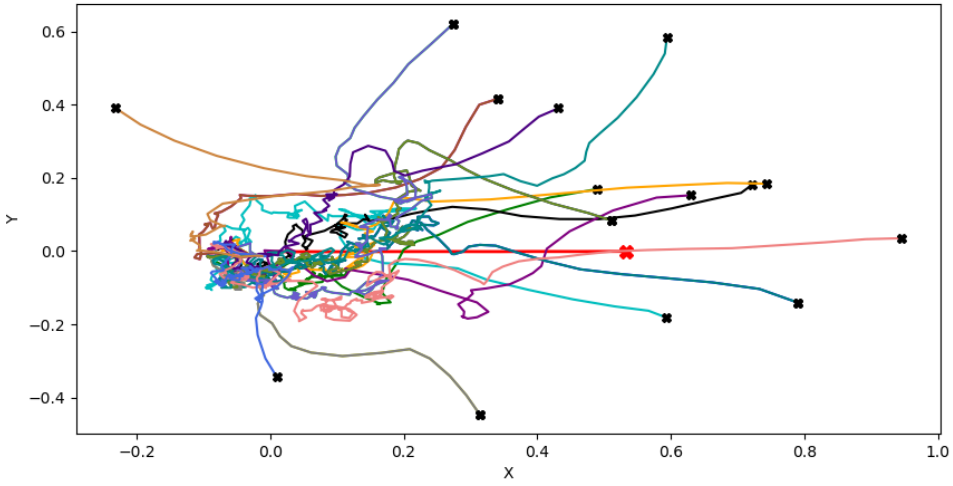


Figure 62 – X-Y view of the trajectory of the root (Hip Yaw) for 20 runs with the learned policy π_{kick4} (in meters). The red line represents the reference trajectory.

The chest Z-axis measurement for 20 runs using the learned policy π_{kick4} is shown in Figure 63. It shows that Marta fell frequently. However, Marta reached the ball on almost every run, as shown in Figure 64. Therefore, given the complex poses for each reference frame, keeping the balance is a more challenging task. The maximum distance covered by the ball after the kick was 2.25 meters on the X-axis, and 1.5 meters on the Y-axis, as shown in Figure 65.

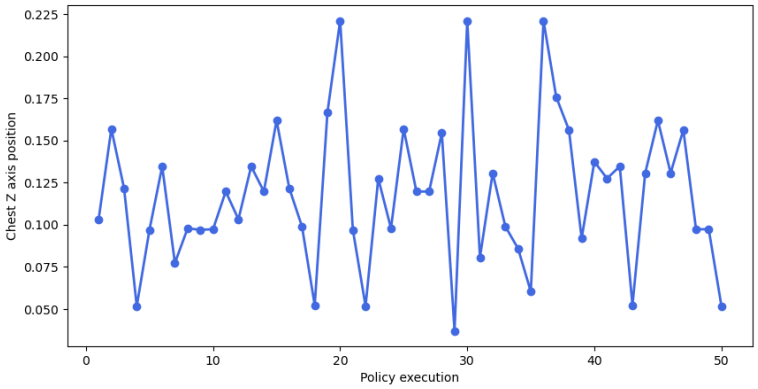


Figure 63 – Final position of the chest z-axis in meters during 50 runs with the learned policy π_{kick4}. The robot was able to stay upright during only 6% of the tests.

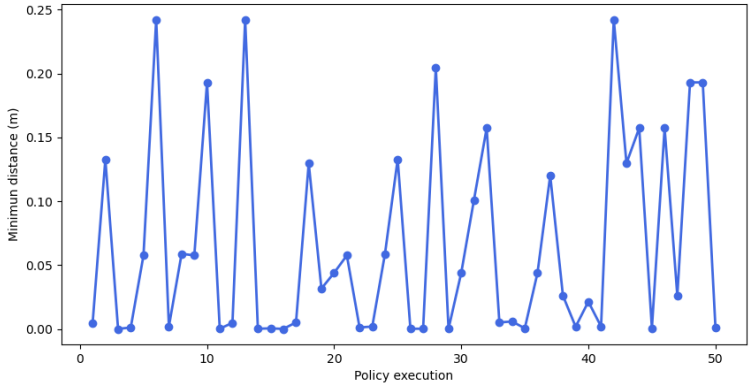


Figure 64 – Minimum distance between the right (kick) foot and the ball position in meters for 50 runs with the learned policy π_{kick4}.

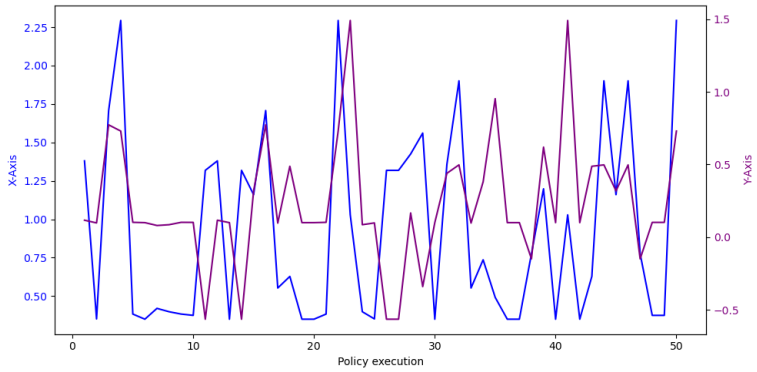


Figure 65 – Distance traveled by the ball (in meters) for 50 runs using the learned policy π_{kick4}.

6.5 Discussion and Limitations

The results confirm that it is feasible to transfer human movements to robots from human video demonstrations with DRL. The robot adapted its physical and dynamic differences to achieve a posture similar to the reference. We also achieved the goal of kicking the ball with the π_{kick3} and π_{kick4} policy.

However, in the method used for training the policies π_{kick1} and π_{kick2} we had to transform the 3D spherical joints movements to 1D revolute joints. Previous experiments have shown that this transformation results in unnatural and often large movements inconsistent with the reference. Then, the movements achieved were not as expected since this method tries to fit the movement into Marta's body, which has different dynamics and dimensions from those of a human. As a result, the robot limited itself to reproduce the movements that brought it the most stability. As a consequence, the robot is limited to exploring the trajectory less.

In the method used for training of policies π_{kick3} and π_{kick4}, the results obtained are promising due to the fact that Marta had to reach her joints at the same coordinates as the resized reference. Joint positions are easier references for robots with body shapes significantly difference from humans than angles. Furthermore, we concluded that the reference frame rescaling was an advantageous strategy in this situation since Marta has a height significantly lower than the adopted reference model.

Our experiments showed that Marta could not follow the same motion cycle as the reference and imitate all the pose frames, even with different time step scenes in CoppeliaSim for all the trained policies. However, the results, especially from the π_{kick3} and π_{kick4} policies, for which Marta obtained a close movement. As a solution, we propose introducing intermediary interpolated frames for each frame in the motion cycle to give the robot more frames to reproduce the trajectory.

Therefore, results confirm that using imitation learning with DRL can transfer human movements to the robots and achieve additional tasks, such as kicking the ball. Table 4 shows an overview of the performed experiments.

Experiments	Time cycle (s)	Time step (s)	Episodes (10^3)	Maximum Reward (10^3)	Maximum Reward average (10^1)	Upright after trained (%)	Average distance traveled by the ball (m)
π_{kick1}	1.24	0.04	70.3	1.04	67.7	58%	0
π_{kick2}	1.24	0.04	66.16	1.02	70	43%	0
π_{kick3}	1.24	0.1	328.08	1.23	6.5	4%	0.57
π_{kick4}	1.24	0.05	92.60	0.29	11.3	6%	1.02

Table 4 – Summary of the results obtained by the robot with the four learned policies with the human motions transferring strategy using DRL.

7 CONCLUSION AND FUTURE WORKS

We presented a framework and investigated methodologies to transfer human movements to robots with DRL from a monocular reference video. The results showed that the proposed approach enabled – at least partially – the reproduction of human movements in the Marta robot with satisfactory posture results, even in cases where the additional goal of kicking the ball was not achieved. As our main contributions, we can highlight:

- The development of a new dataset (*SoccerKicks*) that makes available a collection of dead-ball kick videos (penalty and foul) to provide reference movements for humanoid robots;
  - The Development of a methodology for choosing soccer kicks videos suitable to perform the selected 3D HPE system to obtain a satisfactory result;
  - Providing both the annotation of this database with relevant information for machine learning-based training processes and the evaluation of the motion pattern obtained using some particular metrics;
  - Making available reference movements to be used in the robotic soccer domain;
- The development of an approach to generate the reference motion trajectories for a robot with a body structure significantly different from the reference, particularly considering height;
- The development of two approaches for imitation learning from observations with reinforcement learning, using state-of-the-art techniques from the computer vision field;
- The development of two frameworks that integrate a high fidelity simulator, CoppeliaSim, with the standard robotics toolkit, PyRep, and the RL toolkit, OpenAI Gym and Rlkit;
- The development of a framework that uses the selected dataset to train the realistic humanoid robot Marta with Deep Reinforcement Learning (DRL);
- The proposition and comparison between distinct reward functions and the investigation of techniques that can stimulate more natural and realistic postures from learning.

  For the framework approaches, we highlight:

  - The development of a framework that adapts the reward engineering proposed by (PENG et al., 2018), with the DRL SAC algorithm;
  - The development of a framework improvement that utilizes the position coordinates instead of orientations.

As the DRL modeling does not make any assumptions about the robot model and thus, can be applied to any robot model. However, for it to work for a specific robot model like Marta, the RL environment uses specific information from this robot. Therefore, each new robot model requires its specific environment, that can be coupled to the proposed framework without any further modifications. Although the learned policies appear inappropriate for the actual robot in its current form, they provide valuable references for simulated policies and provide insights into the current design of the robot dynamics that could help in future iterations.

As future work, we suggest the refining of the current policies to fill some of their gaps to optimize learning. We believe this work holds potential to be used as a reference for further learning of different skills based on video recordings of humans performing the most diverse tasks.

BIBLIOGRAPHY

ANDRILUKA, M.; PISHCHULIN, L.; GEHLER, P.; SCHIELE, B. 2d human pose estimation: New benchmark and state of the art analysis. In: *Proceedings of the IEEE Conference on computer Vision and Pattern Recognition*. [S.l.: s.n.], 2014. p. 3686–3693.

ANDRILUKA, M.; ROTH, S.; SCHIELE, B. Pictorial structures revisited: People detection and articulated pose estimation. In: IEEE. *2009 IEEE conference on computer vision and pattern recognition*. [S.l.], 2009. p. 1014–1021.

ARNAB, A.; DOERSCH, C.; ZISSERMAN, A. Exploiting temporal context for 3d human pose estimation in the wild. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. [S.l.: s.n.], 2019. p. 3395–3404.

ARULKUMARAN, K.; DEISENROTH, M. P.; BRUNDAGE, M.; BHARATH, A. A. Deep reinforcement learning: A brief survey. *IEEE Signal Processing Magazine*, IEEE, v. 34, n. 6, p. 26–38, 2017.

BEARMAN, A.; DONG, C. Human pose estimation and activity classification using convolutional neural networks. *CS231n Course Project Reports*, 2015.

BEGAZO, M. F. T. A learning-based model-free controller for decoupled humanoid robot walking. Universidade Estadual Paulista (UNESP), 2020.

BENBRAHIM, H.; FRANKLIN, J. A. Biped dynamic walking using reinforcement learning. *Robotics and Autonomous Systems*, Elsevier, v. 22, n. 3-4, p. 283–302, 1997.

BERKELEY, R. A. L. L. *Rlkit framework*. 2019. Https://github.com/rail- berkeley/rlkit.

BERTASIUS, G.; FEICHTENHOFER, C.; TRAN, D.; SHI, J.; TORRESANI, L. Learning temporal pose estimation from sparsely-labeled videos. In: *Advances in Neural Information Processing Systems*. [S.l.: s.n.], 2019. p. 3027–3038.

BRANTLEY, K.; SUN, W.; HENAFF, M. Disagreement-regularized imitation learning. In: *International Conference on Learning Representations*. [S.l.: s.n.], 2019.

BROCKMAN, G.; CHEUNG, V.; PETTERSSON, L.; SCHNEIDER, J.; SCHULMAN, J.; TANG, J.; ZAREMBA, W. Openai gym. *arXiv preprint arXiv:1606.01540*, 2016.

CAO, Z.; HIDALGO, G.; SIMON, T.; WEI, S.-E.; SHEIKH, Y. Openpose: realtime multi-person 2d pose estimation using part affinity fields. *IEEE transactions on pattern analysis and machine intelligence*, IEEE, v. 43, n. 1, p. 172–186, 2019.

CAO, Z.; SIMON, T.; WEI, S.-E.; SHEIKH, Y. Realtime multi-person 2d pose estimation using part affinity fields. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. [S.l.: s.n.], 2017. p. 7291–7299.

CHEN, T.; FANG, C.; SHEN, X.; ZHU, Y.; CHEN, Z.; LUO, J. Anatomy-aware 3d human pose estimation in videos. *arXiv preprint arXiv:2002.10322*, 2020.

CHEN, Y.; TIAN, Y.; HE, M. Monocular human pose estimation: A survey of deep learning-based methods. *Computer Vision and Image Understanding*, Elsevier, v. 192, p. 102897, 2020.

CHENATTI, S. F.; PREVIATO, G.; TOMAZELA, R.; KOPP, V. G.; BEGAZO, M. F. T.; SALARO, L. G.; ROHMER, E.; COLOMBINI, E. L.; SIMOES, A. da S. Larocs+ unesp team description paper for the ieee humanoid racing 2018. *Latin American Robotics Competition - IEEE Humanoid Racing*, 2018.

CHENATTI, S. F.; PREVIATO, G.; TOMAZELA, R.; KOPP, V. G.; BEGAZO, M. F. T.; SALARO, L. G.; ROHMER, E.; COLOMBINI, E. L.; SIMOES, A. da S. Larocs+ unesp team description paper for the ieee humanoid racing 2018. *Latin American Robotics Competition - IEEE Humanoid Racing*, 2018.

CHENG, Z.; LIU, L.; LIU, A.; SUN, H.; FANG, M.; TAO, D. On the guaranteed almost equivalence between imitation learning from observation and demonstration. *IEEE Transactions on Neural Networks and Learning Systems*, IEEE, 2021.

CORKE, P. *Robotics, vision and control: fundamental algorithms in MATLAB® second, completely revised*. [S.l.]: Springer, 2017. v. 118.

DONG, H.; DING, Z.; ZHANG, S. *Deep Reinforcement Learning: Fundamentals, Research and Applications*. [S.l.]: Springer Nature, 2020.

EDWARDS, A.; SAHNI, H.; SCHROECKER, Y.; ISBELL, C. Imitating latent policies from observation. In: PMLR. *International conference on machine learning*. [S.l.], 2019. p. 1755–1763.

FANG, H.-S.; XIE, S.; TAI, Y.-W.; LU, C. Rmpe: Regional multi-person pose estimation. p. 2334–2343, 2017.

FOUHEY, D. F.; KUO, W.-c.; EFROS, A. A.; MALIK, J. From lifestyle vlogs to everyday interactions. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. [S.l.: s.n.], 2018.

GKIOXARI, G.; TOSHEV, A.; JAITLY, N. Chained predictions using convolutional neural networks. In: SPRINGER. *European Conference on Computer Vision*. [S.l.], 2016. p. 728–743.

GŁOWIŃSKI, S.; KRZYŻYŃSKI, T. An inverse kinematic algorithm for the human leg. *Journal of theoretical and Applied Mechanics*, v. 54, n. 1, p. 53–61, 2016.

GONÇALVES, R. F. *Deep Reinforcement Learning Application in Humanoid Robots Locomotion*. 103 f. Dissertação (Mestrado) — Universidade Estadual de Campinas, Instituto de Computação, Campinas,SP, 2021.

GOODFELLOW, I.; POUGET-ABADIE, J.; MIRZA, M.; XU, B.; WARDE-FARLEY, D.; OZAIR, S.; COURVILLE, A.; BENGIO, Y. Generative adversarial nets. In: *Advances in neural information processing systems*. [S.l.: s.n.], 2014. p. 2672–2680.

GOSWAMI, A.; VADAKKEPAT, P. *Humanoid robotics: A reference*. [S.l.]: Springer, 2019.

GRONDMAN, I.; BUSONIU, L.; LOPES, G. A.; BABUSKA, R. A survey of actor-critic reinforcement learning: Standard and natural policy gradients. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, IEEE, v. 42, n. 6, p. 1291–1307, 2012.

GUPTA, A.; DEVIN, C.; LIU, Y.; ABBEEL, P.; LEVINE, S. Learning invariant feature spaces to transfer skills with reinforcement learning. *arXiv preprint arXiv:1703.02949*, 2017.

HAARNOJA, T.; ZHOU, A.; HARTIKAINEN, K.; TUCKER, G.; HA, S.; TAN, J.; KUMAR, V.; ZHU, H.; GUPTA, A.; ABBEEL, P. et al. Soft actor-critic algorithms and applications. *arXiv preprint arXiv:1812.05905*, 2018.

HE, K.; ZHANG, X.; REN, S.; SUN, J. Identity mappings in deep residual networks. In: SPRINGER. *European conference on computer vision*. [S.l.], 2016. p. 630–645.

HO, J.; ERMON, S. Generative adversarial imitation learning. In: *Advances in neural information processing systems*. [S.l.: s.n.], 2016. p. 4565–4573.

HUDSON, E.; WARNELL, G.; TORABI, F.; STONE, P. Skeletal feature compensation for imitation learning with embodiment mismatch. *arXiv preprint arXiv:2104.07810*, 2021.

HUSSEIN, A.; GABER, M. M.; ELYAN, E.; JAYNE, C. Imitation learning: A survey of learning methods. *ACM Computing Surveys (CSUR)*, ACM New York, NY, USA, v. 50, n. 2, p. 1–35, 2017.

IONESCU, C.; PAPAVA, D.; OLARU, V.; SMINCHISESCU, C. Human3. 6m: Large scale datasets and predictive methods for 3d human sensing in natural environments. *IEEE transactions on pattern analysis and machine intelligence*, IEEE, v. 36, n. 7, p. 1325–1339, 2013.

JAMES, S.; FREESE, M.; DAVISON, A. J. Pyrep: Bringing v-rep to deep robot learning. *arXiv preprint arXiv:1906.11176*, 2019.

JAZAR, R. N. *Theory of applied robotics: kinematics, dynamics, and control*. [S.l.]: Springer Science & Business Media, 2010.

JOHNSON, S.; EVERINGHAM, M. Clustered pose and nonlinear appearance models for human pose estimation. In: *Proceedings of the British Machine Vision Conference*. [S.l.]: BMVA Press, 2010. p. 12.1–12.11.

KANAZAWA, A.; BLACK, M. J.; JACOBS, D. W.; MALIK, J. End-to-end recovery of human shape and pose. In: *Computer Vision and Pattern Regognition (CVPR)*. [S.l.: s.n.], 2018.

KANAZAWA, A.; ZHANG, J. Y.; FELSEN, P.; MALIK, J. Learning 3d human dynamics from video. In: *Computer Vision and Pattern Recognition (CVPR)*. [S.l.: s.n.], 2019.

KANAZAWA, A.; ZHANG, J. Y.; FELSEN, P.; MALIK, J. *InstaVariety Dataset*. 2021.

KANAZAWA, A.; ZHANG, J. Y.; FELSEN, P.; MALIK, J. *VLOG-people Dataset*. 2021.

KIM, D.; LEE, J.; SENTIS, L. Robust dynamic locomotion via reinforcement learning and novel whole body controller. *arXiv preprint arXiv:1708.02205*, 2017.

KIMURA, D.; CHAUDHURY, S.; TACHIBANA, R.; DASGUPTA, S. Internal model from observations for reward shaping. *arXiv preprint arXiv:1806.01267*, 2018.

KISSOS, I.; FRITZ, L.; GOLDMAN, M.; MEIR, O.; OKS, E.; KLIGER, M. Beyond weak perspective for monocular 3d human pose estimation. In: SPRINGER. *European Conference on Computer Vision*. [S.l.], 2020. p. 541–554.

KUEHNE, H.; JHUANG, H.; GARROTE, E.; POGGIO, T.; SERRE, T. Hmdb: a large video database for human motion recognition. In: IEEE. *2011 International Conference on Computer Vision*. [S.l.], 2011. p. 2556–2563.

LASKEY, M.; LEE, J.; FOX, R.; DRAGAN, A.; GOLDBERG, K. Dart: Noise injection for robust imitation learning. *arXiv preprint arXiv:1703.09327*, 2017.

LESSA, N. M.; COLOMBINI, E. L.; SIMÕES, A. D. S. Soccerkicks: a dataset of 3d dead ball kicks reference movements for humanoid robots. In: IEEE. *2021 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*. [S.l.], 2021. p. 3472–3478.

LESSA, N. M.; COLOMBINI, E. L.; SIMÕES, A. d. S. *SoccerKicks dataset*. 2021. Available at <https://github.com/larocs/SoccerKicks>, Accessed: 18.06.2021.

LI, H.; YANG, W.; LIAO, Q. Temporal feature enhancing network for human pose estimation in videos. In: IEEE. *2019 IEEE International Conference on Image Processing (ICIP)*. [S.l.], 2019. p. 579–583.

LI, J.; WANG, C.; ZHU, H.; MAO, Y.; FANG, H.-S.; LU, C. Crowdpose: Efficient crowded scenes pose estimation and a new benchmark. *arXiv preprint arXiv:1812.00324*, 2018.

LI, Y.-L.; XU, L.; LIU, X.; HUANG, X.; XU, Y.; CHEN, M.; MA, Z.; WANG, S.; FANG, H.-S.; LU, C. Hake: Human activity knowledge engine. *arXiv preprint arXiv:1904.06539*, 2019.

LI, Z.; WANG, X.; WANG, F.; JIANG, P. On boosting single-frame 3d human pose estimation via monocular videos. In: *Proceedings of the IEEE International Conference on Computer Vision*. [S.l.: s.n.], 2019. p. 2192–2201.

LIM, S.; OH, S.; KIM, K. I. Balance control for biped walking robots using only zero-moment-point position signal. *Electronics letters*, IET, v. 48, n. 1, p. 19–20, 2012.

LIN, T.-Y.; MAIRE, M.; BELONGIE, S.; HAYS, J.; PERONA, P.; RAMANAN, D.; DOLLÁR, P.; ZITNICK, C. L. Microsoft coco: Common objects in context. In: SPRINGER. *European conference on computer vision*. [S.l.], 2014. p. 740–755.

LIU, C.; NING, J.; CHEN, Q. Dynamic walking control of humanoid robots combining linear inverted pendulum mode with parameter optimization. *International Journal of Advanced Robotic Systems*, SAGE Publications Sage UK: London, England, v. 15, n. 1, p. 1729881417749672, 2018.

LIU, Y.; CHEN, J. Posepropagationnet: Towards accurate and efficient pose estimation in videos. *IEEE Access*, IEEE, 2020.

LIU, Y.; GUPTA, A.; ABBEEL, P.; LEVINE, S. Imitation from observation: Learning to imitate behaviors from raw video via context translation. In: IEEE. *2018 IEEE International Conference on Robotics and Automation (ICRA)*. [S.l.], 2018. p. 1118–1125.

LOPER, M.; MAHMOOD, N.; ROMERO, J.; PONS-MOLL, G.; BLACK, M. J. SMPL: A skinned multi-person linear model. *ACM Trans. Graphics (Proc. SIGGRAPH Asia)*, ACM, v. 34, n. 6, p. 248:1–248:16, out. 2015.

MACCONAILL, M. A. *Joint*. 2021. Available at <https://www.britannica.com/science/joint-skeleton>, Accessed: 06.11.2021.

MAHMOOD, N.; GHORBANI, N.; TROJE, N. F.; PONS-MOLL, G.; BLACK, M. J. Amass: Archive of motion capture as surface shapes. In: *Proceedings of the IEEE International Conference on Computer Vision*. [S.l.: s.n.], 2019. p. 5442–5451.

MARCARD, T. von; HENSCHEL, R.; BLACK, M. J.; ROSENHAHN, B.; PONS-MOLL, G. Recovering accurate 3d human pose in the wild using imus and a moving camera. In: *Proceedings of the European Conference on Computer Vision (ECCV)*. [S.l.: s.n.], 2018. p. 601–617.

MARSLAND, S. *Machine learning: an algorithmic perspective*. [S.l.]: CRC press, 2015.

MEREL, J.; TASSA, Y.; TB, D.; SRINIVASAN, S.; LEMMON, J.; WANG, Z.; WAYNE, G.; HEESS, N. Learning human behaviors from motion capture by adversarial imitation. *arXiv preprint arXiv:1707.02201*, 2017.

MITCHELL, T. M. et al. *Machine learning. 1997*. [S.l.: s.n.], 1997.

MNIH, V.; KAVUKCUOGLU, K.; SILVER, D.; RUSU, A. A.; VENESS, J.; BELLEMARE, M. G.; GRAVES, A.; RIEDMILLER, M.; FIDJELAND, A. K.; OSTROVSKI, G. et al. Human-level control through deep reinforcement learning. *nature*, Nature Publishing Group, v. 518, n. 7540, p. 529–533, 2015.

NAIR, A.; CHEN, D.; AGRAWAL, P.; ISOLA, P.; ABBEEL, P.; MALIK, J.; LEVINE, S. Combining self-supervised learning and imitation for vision-based rope manipulation. In: IEEE. *2017 IEEE international conference on robotics and automation (ICRA)*. [S.l.], 2017. p. 2146–2153.

OMPICO, C.; BUGTAI, N.; MUNSAYAC, F. Recent developments on social robots and imitation learning for robotic therapy. In: IOP PUBLISHING. *Journal of Physics: Conference Series*. [S.l.], 2021. v. 2071, n. 1, p. 012021.

OSA, T.; PAJARINEN, J.; NEUMANN, G.; BAGNELL, J. A.; ABBEEL, P.; PETERS, J. An algorithmic perspective on imitation learning. *arXiv preprint arXiv:1811.06711*, 2018.

PASZKE, A.; GROSS, S.; CHINTALA, S.; CHANAN, G.; YANG, E.; DEVITO, Z.; LIN, Z.; DESMAISON, A.; ANTIGA, L.; LERER, A. Automatic differentiation in pytorch. 2017.

PAVLLO, D.; FEICHTENHOFER, C.; GRANGIER, D.; AULI, M. 3d human pose estimation in video with temporal convolutions and semi-supervised training. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. [S.l.: s.n.], 2019. p. 7753–7762.

PEDRYCZ, W.; CHEN, S.-M. *Deep Learning: Algorithms and Applications*. [S.l.]: Springer, 2020.

PEDRYCZ, W.; CHEN, S.-M. *Deep Learning: Concepts and Architectures*. [S.l.]: Springer, 2020.

PENG, X. B.; ABBEEL, P.; LEVINE, S.; PANNE, M. van de. Deepmimic: Example-guided deep reinforcement learning of physics-based character skills. *ACM Trans. Graph.*, ACM, New York, NY, USA, v. 37, n. 4, p. 143:1–143:14, jul. 2018. ISSN 0730-0301.

PENG, X. B.; COUMANS, E.; ZHANG, T.; LEE, T.-W.; TAN, J.; LEVINE, S. Learning agile robotic locomotion skills by imitating animals. *arXiv preprint arXiv:2004.00784*, 2020.

PENG, X. B.; COUMANS, E.; ZHANG, T.; LEE, T.-W.; TAN, J.; LEVINE, S. *Motion Imitation*. 2020. Available at <https://github.com/erwincoumans/motion_imitation>, Accessed: 14.05.2022.

PENG, X. B.; KANAZAWA, A.; MALIK, J.; ABBEEL, P.; LEVINE, S. Sfv: Reinforcement learning of physical skills from videos. *ACM Transactions on Graphics (TOG)*, ACM New York, NY, USA, v. 37, n. 6, p. 1–14, 2018.

PENG, X. B.; MA, Z.; ABBEEL, P.; LEVINE, S.; KANAZAWA, A. Amp: Adversarial motion priors for stylized physics-based character control. *ACM Trans. Graph.*, ACM, New York, NY, USA, v. 40, n. 4, jul. 2021.

PENG, X. B. J.; KANAZAWA, A. *Learning Acrobatics by Watching YouTube*. 2018. Available at <https://bair.berkeley.edu/blog/2018/10/09/sfv/>, Accessed: 02.09.2021.

PFISTER, T.; CHARLES, J.; ZISSERMAN, A. Flowing convnets for human pose estimation in videos. In: *Proceedings of the IEEE International Conference on Computer Vision*. [S.l.: s.n.], 2015. p. 1913–1921.

PISHCHULIN, L.; ANDRILUKA, M.; GEHLER, P.; SCHIELE, B. Poselet conditioned pictorial structures. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. [S.l.: s.n.], 2013. p. 588–595.

PYBULLET. *Inverse Kinect and Reference Rendering*. 2019. Available at <https://github.com/bulletphysics/bullet3/tree/master/examples/pybullet/gym/pybullet_envs/deep_mimic/mocap>, Accessed: 14.05.2022.

REDSHIFT-LABS. *UM7 Orientation Sensor*. Available at <http://www.chrobotics.com/shop/um7-orientation-sensor>, Accessed: 21.10.2021.

ROBOTICS, C. *Virtual Robot Experimentation Platform - USER MANUAL*. Available at <http://www.coppeliarobotics.com/helpFiles/index.htmle>, Accessed: 21.10.2021.

RUSSEL, S.; P, N. *Artificial intelligence. A modern approach*. [S.l.]: Prentice Hall, 2013.

SCHULMAN, J.; WOLSKI, F.; DHARIWAL, P.; RADFORD, A.; KLIMOV, O. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

SERMANET, P.; LYNCH, C.; CHEBOTAR, Y.; HSU, J.; JANG, E.; SCHAAL, S.; LEVINE, S.; BRAIN, G. Time-contrastive networks: Self-supervised learning from video. In: IEEE. *2018 IEEE International Conference on Robotics and Automation (ICRA)*. [S.l.], 2018. p. 1134–1141.

SEWAK, M. *Deep Reinforcement Learning: Frontiers of Artificial Intelligence*. [S.l.]: Springer, 2019.

SICILIANO, B.; KHATIB, O. *Springer handbook of robotics*. [S.l.]: Springer, 2016.

SILVER, D.; HUANG, A.; MADDISON, C. J.; GUEZ, A.; SIFRE, L.; DRIESSCHE, G. V. D.; SCHRITTWIESER, J.; ANTONOGLOU, I.; PANNEERSHELVAM, V.; LANCTOT, M. et al. Mastering the game of go with deep neural networks and tree search. *nature*, Nature Publishing Group, v. 529, n. 7587, p. 484–489, 2016.

SIMON, T.; JOO, H.; MATTHEWS, I.; SHEIKH, Y. Hand keypoint detection in single images using multiview bootstrapping. In: *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*. [S.l.: s.n.], 2017. p. 1145–1153.

SOARES, Y. C. P. et al. *Deep reinforcement learning for bipedal locomotion: Aprendizado por reforço profundo para locomoção bípede*. Tese (Doutorado) — Universidade Estadual de Campinas, Instituto de Computação, 2020.

SUTTON, R. S.; BARTO, A. G. *Reinforcement learning: An introduction. Second edition, in progress.* [S.l.]: MIT press, 2015.

SUTTON, R. S.; BARTO, A. G. *Reinforcement learning: An introduction*. [S.l.]: MIT press, 2018.

TEKSCAN. *FlexiForce Standard Model A201*. Available at <https://cdn.sparkfun.com/datasheets/Sensors/ForceFlex/FLX-A201-A.pdf>, Accessed: 21.10.2021.

TOMAZELA, R. M. *A combined model-based planning and model-free reinforcement learning approach for biped locomotion: Uma abordagem combinada de planejamento baseado em modelo e aprendizado por reforço para locomoção bípede*. 87 f. Dissertação (Mestrado) — Universidade Estadual de Campinas, Instituto de Computação, Campinas,SP, 2019.

TORABI, F.; WARNELL, G.; STONE, P. Recent advances in imitation learning from observation. *arXiv preprint arXiv:1905.13566*, 2019.

TOSHEV, A.; SZEGEDY, C. Deeppose: Human pose estimation via deep neural networks. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. [S.l.: s.n.], 2014. p. 1653–1660.

VONDRAK, M.; SIGAL, L.; HODGINS, J.; JENKINS, O. Video-based 3d motion capture through biped control. *ACM Transactions On Graphics (TOG)*, ACM New York, NY, USA, v. 31, n. 4, p. 1–12, 2012.

WANG, F.; LI, Y. Beyond physical connections: Tree models in human pose estimation. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. [S.l.: s.n.], 2013. p. 596–603.

WANG, G.; WU, Q. J. *Guide to three dimensional structure and motion factorization*. [S.l.]: Springer, 2011.

WATKINS, C. J.; DAYAN, P. Q-learning. *Machine learning*, Springer, v. 8, n. 3-4, p. 279–292, 1992.

WEI, S.-E.; RAMAKRISHNA, V.; KANADE, T.; SHEIKH, Y. Convolutional pose machines. In: *CVPR*. [S.l.: s.n.], 2016.

WU, M.; GAO, Y.; JUNG, A.; ZHANG, Q.; DU, S. The actor-dueling-critic method for reinforcement learning. *Sensors*, Multidisciplinary Digital Publishing Institute, v. 19, n. 7, p. 1547, 2019.

XIE, Z.; CLARY, P.; DAO, J.; MORAIS, P.; HURST, J.; PANNE, M. van de. Iterative reinforcement learning based design of dynamic locomotion skills for cassie. *arXiv preprint arXiv:1903.09537*, 2019.

XIE, Z.; CLARY, P.; DAO, J.; MORAIS, P.; HURST, J.; PANNE, M. van de. Learning locomotion skills for cassie: Iterative design and sim-to-real. In: *Proc. Conference on Robot Learning (CORL 2019)*. [S.l.: s.n.], 2019. v. 4.

XIU, Y.; LI, J.; WANG, H.; FANG, Y.; LU, C. Pose Flow: Efficient online pose tracking. In: *BMVC*. [S.l.: s.n.], 2018.

YANG, C.; YUAN, K.; HENG, S.; KOMURA, T.; LI, Z. Learning natural locomotion behaviors for humanoid robots using human bias. *IEEE Robotics and Automation Letters*, IEEE, v. 5, n. 2, p. 2610–2617, 2020.

YANG, Y.; RAMANAN, D. Articulated pose estimation with flexible mixtures-of-parts. In: IEEE. *CVPR 2011*. [S.l.], 2011. p. 1385–1392.

YU, T.; FINN, C. *One-Shot Imitation from Watching Videos*. 2018. Available at <https://bair.berkeley.edu/blog/2018/06/28/daml/>, Accessed: 10.05.2022.

YU, W. *LEARNING TO WALK USING DEEP REINFORCEMENT LEARNING AND TRANSFER LEARNING*. Tese (Doutorado) — Georgia Institute of Technology, 2020.

YU, W.; KUMAR, V. C.; TURK, G.; LIU, C. K. Sim-to-real transfer for biped locomotion. *arXiv preprint arXiv:1903.01390*, 2019.

ZAHRA, O.; TOLU, S.; ZHOU, P.; DUAN, A.; NAVARRO-ALARCON, D. A bio-inspired mechanism for learning robot motion from mirrored human demonstrations. *Frontiers in Neurorobotics*, Frontiers Media SA, v. 16, 2022.

ZHANG, D.; GUO, G.; HUANG, D.; HAN, J. Poseflow: A deep motion representation for understanding human behaviors in videos. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. [S.l.: s.n.], 2018. p. 6762–6770.

ZHANG, H.; LIU, Y.; ZHOU, W. Deep adversarial imitation learning of locomotion skills from one-shot video demonstration. In: IEEE. *2019 IEEE 9th Annual International Conference on CYBER Technology in Automation, Control, and Intelligent Systems (CYBER)*. [S.l.], 2019. p. 1257–1261.

ZHOU, D.; FANG, J.; SONG, X.; GUAN, C.; YIN, J.; DAI, Y.; YANG, R. Iou loss for 2d/3d object detection. In: IEEE. *2019 International Conference on 3D Vision (3DV)*. [S.l.], 2019. p. 85–94.

ZIEBART, B. D.; MAAS, A. L.; BAGNELL, J. A.; DEY, A. K. Maximum entropy inverse reinforcement learning. In: CHICAGO, IL, USA. *Aaai*. [S.l.], 2008. v. 8, p. 1433–1438.