LEARNING EXPRESSIVE QUADRUPEDAL LOCOMOTION GUIDED BY KINEMATIC TRAJECTORY GENERATOR

A Master thesis
Presented to
The Academic Faculty

By

Arnaud Klipfel

In Partial Fulfillment
of the Requirements for the Master degree in the
school of Electrical and Computer engineering
College of engineering

Georgia Institute of Technology

December 2022

LEARNING EXPRESSIVE QUADRUPEDAL LOCOMOTION GUIDED BY KINEMATIC TRAJECTORY GENERATOR

Thesis committee:

Dr. Seth Hutchinson, Advisor
School of Interactive Computing
Georgia Institute of Technology

Dr. Zsolt Kira
School of Interactive Computing
Georgia Institute of Technology

Dr. Sehoon Ha, Co-Advisor
School of Interactive Computing
Georgia Institute of Technology

Dr. Ye Zhao
School of Mechanical Engineering
Georgia Institute of Technology

Date approved: December 12th 2022

One must have chaos in oneself to give birth to a dancing star.

Friedrich Nietzsche

For my Family, Science, a better world and the beauty of life.

ACKNOWLEDGMENTS

I would like to thank the members of my thesis committee for their help in assessing and commenting my work. I would like to give a special thank to Dr. Seth Hutchinson and Dr. Sehoon Ha for their continuous support, creative inputs in my work. A special thank to Gabriel, Mehdi and Maks for their friendship.

TABLE OF CONTENTS

LIST OF FIGURES

SUMMARY

The goal of this thesis was to study and implement Deep Reinforcement Learning (DRL) algorithms applied to quadrupedal legged locomotion. More model-based approaches can also be considered to solve the problem (e.g. [1]), but this work focuses on using DRL techniques as these approaches showcased incredible state-of-the-art results [2, 3, 4]. More precisely, the goal was to learn a policy exhibiting a wide range of locomotion skills, i.e. walking forward, turning, squatting, turning in place, and jumping for instance, and deploy the policy in the real world. Most locomotion policies deployed in the real world only exhibit single locomotion skills and can only solve very narrow locomotion tasks as a result. A common way to expand the pool of available locomotion skills is to train narrow policies for different skills and then combine them [5].

The first part of the thesis focused on reproducing the state of the art in robust walking policies, which has been well researched and even solved recently [3]. In that case, the focus is to produce a walking gait (no jump or dynamical motions) to overcome challenging terrains such as those that can be found in the wild. Methods using reward shaping were tested first. In Reinforcement Learning (RL) the reward is the objective function, goal that is being maximized. A reference-based method, the PMTG [6] was reproduced and presented in section 2.2, followed by a reference-free approach based on [2] and presented in section 2.3. Reference-free approaches are equivalent to methods that do not use any guiding signals for the learning except the reward, also known as learning from ”scratch”, i.e. no prior knowledge is assumed on the structure of the actions, in that case, the joint motor angles. Reference-based methods use a family of functions that they modulate, i.e. these

techniques learn the parameters of these functions. Ultimately, the goal was to modify these approaches to produce a jumping motion and later a vast repertoire of motions. Yet, on the one hand, specifying the diverse behaviors solely through reward signals is challenging, not intuitive, and does not lead to natural motions. On the other hand, finding a structure for the motor angles to generate a diverse set of behavior is also challenging. The main issue in the generation of a diverse set of motions is their specifications.

Kinematic references (i.e. recorded behaviors, motion capture data, synthetically generated reference motions) can be used to specify the core locomotion skills. The last part deals with the design of an expressive policy (exhibiting different locomotion skills) based on kinematic references generated by a Neural Network ([7], presented in chapter 3), which is itself based on dog motion capture data. It is possible to break free from the exclusive reward specification of the behavior. The problem of the reward shaping is then only concerned with the kinematic reference tracking.

The main conclusion from this thesis is that in order to generate much more agile locomotion skills such as turning in place, and jumping, for instance, using motion clips describing the motion is much more intuitive than specifying the motion directly in the reward. For simple behaviors such as walking approaches using pure reward shaping exhibit good results, as presented in chapter 2, but when it comes to specifying a jumping behavior for instance, and learning it, motion clips provide an intuitive mean. chapter 3 shows a sample of agile locomotion skills learned by a single policy through motion clip data and using DRL as in [4]. Another conclusion is when exclusively considering walking gaits to compare the three tested approaches, walking gaits obtained with the motion clip approach presented in chapter 3 look more natural.

Supplementary materials for this thesis such as video, code are available as explained in Appendix A.

CHAPTER 1

INTRODUCTION

Producing systems that could imitate the motions of animals is not a recent phenomenon. As early as 500 BC, Archytas, a Greek philosopher, designed a flying pigeon [8]. Locomotion has been studied for centuries, trying to understand the biology behind the different gaits of legged animals (e.g. trotting, walking, galloping) [9, 10]. It is only in the sixties that engineers started designing legged robotic systems. The first one was "Phony Pony". Since then plenty of commercial bipedal and quadrupedal robotic platforms have been designed [11, 12, 13].

Marc Raibert was one of the first to lay down the modern foundations of the control of legged robots [14]. Dynamically stable locomotion controllers were designed using control theory and modeling a legged robot as a non linear dynamical system, and using heuristics such as foot placements during swing phases (aerial phases of locomotion) in combination with position controllers. Assuming the different bodies of the legged system are rigid bodies, i.e. the different limbs and the torso cannot be deformed by any applied forces, the dynamical equations of a legged system (composed of a torso and N limbs) can be written as [15]:

$$\begin{aligned} \mathbf{M}_u(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{h}_u(\mathbf{q}, \dot{\mathbf{q}}) &= \mathbf{J}_u(\mathbf{q})^T\mathbf{f} \\ \mathbf{M}_a(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{h}_a(\mathbf{q}, \dot{\mathbf{q}}) &= \boldsymbol{\tau} + \mathbf{J}_a(\mathbf{q})^T\mathbf{f} \end{aligned} \tag{1.1}$$

The first equation is the torso equation, which is the unactuacted part of the system and the second equation describes the dynamics of the limbs, which are actuated at the joints with motors. $\mathbf{M}$ denotes the inertia matrix linked to the acceleration of the system, and $\mathbf{h}$ a non linear term linking the generalized coordinates $\mathbf{q}$ and the generalized velocities $\dot{\mathbf{q}}$. For a legged system, the generalized coordinates contain the torso position and orientation in the world frame and the N_j controllable joint angles. For common quadrupedal systems such

as commercial platforms, $\mathbf{q} \in \mathbb{R}^{18}$, and $N_j = 12$. $\mathbf{J}$ denotes the Jacobian matrix of the system, projecting the contact forces $\mathbf{f}$ or ground reaction forces (GRF) to the torso or limb frames, τ is the actuation torque vector.

The locomotion is divided into different phases, characterized by the phase of each leg: in the swing phase the foot is not in contact with the ground, and in the stance phase the foot is in contact with the ground. A foot can be in a swing or stance phase. Heuristics are analytical relations found empirically, i.e. through experiments, and describe, and approximate certain desired behaviors. Heuristics are commonly used in Reinforcement Learning to guide the learning or to reduce the space of actions to learn. Raibert heuristics are still used in some works to compute the foot or end-effector trajectories in a swing or aerial phase as they are simple analytical relations [16]. [17, 16] used a Raibert heuristic in order to produce the next desired foot placement $p_{d,i} \in \mathbb{R}^3$, for the foot $i \in [1; 4]$ based on the desired body velocity $\dot{p}_{d,\text{ body}} \in \mathbb{R}^3$, the current body velocity $\dot{p}_{\text{body}}$, and pre-defined foot position $p_{0,i}$, as follows:

$$p_{d,i} = p_{0,i} + k\left(\dot{p}_{\text{body}} - \dot{p}_{d,\text{ body}}\right)$$

where $k > 0$ is a control gain that has to be tuned. If the robot is too slow, this heuristic will shorten the robot stride (smaller steps), which will result in a faster motion of the legs, and a faster gait so the robot can keep up with the desired body velocity. As in [16, 17] position controllers are then producing the required torques for each actuator or joint of the leg.

Recent advances in trajectory optimization and dynamic programming [15] enabled to cast the locomotion gait generation (foot placement, joint position, floating base state) as a linear or non linear program. Using different sets of approximations on the dynamics of the robotic system Equation 1.1, the dynamical equations and constraints on the locomotion (foot placements, swing and stance phase) can become convex and even linear, or quadratic which makes the optimization faster. Using optimization techniques such as MPC (Model Predictive Control) and QP (Quadratic Programming) locomotion gaits were generated for

diverse robots [18, 19, 1, 20, 21].

The main disadvantage of model-based techniques, is the use of a dynamical model that does not account for the full dynamic of the legged system, long optimization times, and hand-engineered heuristics used to make the optimization process faster. Deep Reinforcement Learning (DRL) was first applied to computer animation in the field of locomotion and achieved impressive results in the generation of very acrobatic motions such as jumps, cartwheels [4, 22, 23]. Yet, these works were confined to simulation. Recently, [3, 24, 25, 26] pushed the limits of DRL techniques by deploying policies learned in simulation on wild terrain showcasing incredible robustness and adaptability.

Some works have tried to get the best of both worlds by combining trajectory-based and DRL techniques [16, 27]. Trajectory-optimization based solutions do not require long training times, can work online, and can be proven to converge in some cases, while learning-based techniques using DRL can truly push the limits of learned behaviors, producing more agile and natural motions. DRL techniques also enable the extraction of meaningful representations and fuse diverse sensory inputs for both exteroception (vision) and proprioception (encoders, Inertia Measurement Unit).

This work focuses on the study of DRL-based locomotion techniques. In order to reduce the training times a common technique is to use a set of parameters that the policy can learn instead of learning the low-level motor space (joint target positions or motor angles for all joints). The Policies Modulating Trajectory Generator (PMTG) [6, 3] learns the parameters of a family of functions to this end. Other methods solely use reward shaping to specify the motions and learn low-level motor actions directly [2]. Finally, this work tackles the synthesis of an expressive locomotion policy, which could generate a wide set of motions. Imitation Learning as in [4] is used to track motion clips (successive states of a moving agent).

In all the experiments presented here the problem of learning actions for legged locomotion is formulated as a reinforcement learning problem where the goal is to learn a

policy $\pi(a|o)$, where a is the learned action vector, and o the observation vector. As is often the case in the real world, the learning problem can be partially observable, i.e. it is not possible to recover the entire robot state s from the observations the policy has. Observations might also contain more than just the robot state, for instance, high-level commands such as a target velocity or environment information. The robot state s are the generalized coordinates of the robot, Center of Mass (CoM) pose, and joint positions. In Reinforcement learning (RL), the transitions are formulated as a Markov Decision Process, where at time t an action is sampled from the policy $a_t \sim \pi(a_t|o_t)$. The policy is defined as a multivariate Gaussian distribution. Given the action a_t and the current state of the agent s_t the agent is then transitioned to state s_{t+1} following a transition model $\mathcal{F}$, as

$$s_{t+1} = \mathcal{F}(a_t, s_t, e_t)$$

e_t is the state of the environment that will impact the evolution of the agent based on its dynamics. Computing the action a_t is done by providing a reward judging the relevance of the taken action in the light of an objective function called the reward, $r_t = r(s_t, a_t, s_{t+1})$. For an entire trajectory of actions τ, the objective is written as,

$$J(\pi) = \mathbb{E}_{\tau \sim p(\tau|\pi)} \left[\sum_{t=0}^{T-1} \gamma^t r_t \right]$$

where $T \in \mathbb{R}$ is the final step of the current episode, and $\gamma \in [0; 1[$ is the discount factor that weights the impact of future possible outcomes due to action a_t. $p(\tau \mid \pi)$ is the probability of having the action trajectory τ given the learned policy π. To solve the Reinforcement Learning problem the Reinforcement Learning algorithm PPO (Proximal Policy Optimization) [28] was used.

This work focused on quadrupedal locomotion, and mainly on the Unitree [12] platforms A1 and Aliengo, which have a total of 12 joints or 12 DC motors (electrical motors) that can be controlled. For each joint or DC motor one angle is controllable. Figure 1.1

presents the morphology of the robot. The Raisim [29] physics engine and simulator are used for the rollouts, i.e. sampling from the policy and simulation of the action outcome in an environment.

Figure 1.1: Aliengo robot model from Unitree [12]. Each leg has a hip (joint closest to the base), thigh, and calf (closest to the foot) motor angle that can be controlled. For the remainder of the thesis the order in which joints are considered is: front right, front left, rear right and rear left leg.

CHAPTER 2
LEARNING TO WALK THROUGH REWARD SHAPING

2.1 Introduction

The first stage of the thesis focused on reproducing the state-of-the-art in robust walking policies. Reference-based and reference-free approaches have been implemented and trained to obtain a smooth and natural walking policy. As in [6] a reference-based approach consists in learning parameters of a function, while as in [2] a reference-free approach consists in learning directly in the joint space. The goal was then to attempt adapting these approaches to learn more complex locomotion skills.

2.2 Reference-based policy learning

2.2.1 Technical details

The PMTG consists of learning joint residuals, i.e. small variations, around a joint reference trajectory. Different task spaces could be considered, [30] defines the trajectory in the task or cartesian space. For a greater degree of flexibility over the motion, the joint space was chosen over the cartesian space. The agent learns the residuals around the reference trajectory and learns the parameters of the reference trajectory, which enables more flexibility in the motion and allows the policy to adapt the reference based on the observations. The PMTG can be seen a guided learning where the reference trajectory or the family of curves the policy learns to modulate acts like a scaffold used for more structured learning than learning from scratch (only from reward signals) as in [2].

The action space $\mathcal{A}$ consists of the 12 motor angles or joint position residuals $\tilde{\mathbf{q}} \in \mathbb{R}^{12}$, the gait frequency f, and the thigh and calf amplitudes residuals $\tilde{A}$ for each leg. The dimension of the action space is $dim(\mathcal{A}) = 21$. The final joint position target or the final

actions applied to the robot can be written as:

$$\mathbf{q}_\pi = \mathbf{q}_{tg} + \tilde{\mathbf{q}} \tag{2.1}$$

where $\mathbf{q}_{tg}$ is the reference trajectory that the policy can learn the parameters of (amplitudes and frequency), and $\tilde{\mathbf{q}}$ denotes the joint residuals. The joint residual action bound or absolute maximum joint residual is chosen to be $1.5\ rad$ as it enables to have the required range to generate a walking gait. The reference trajectory for the thigh and calf joints was chosen as:

$$\theta_{tg}(t) = (A + \tilde{A})cos(2\pi ft + \Phi_0) + A + \theta_{min} \tag{2.2}$$

$$A = \frac{(\theta_{max} - \theta_{min})}{2}$$

The learned parameters are $\tilde{A}$ the amplitude residuals for a total of 8, and the gait frequency f, which is the same for each leg. The other parameters of the trajectory generator are chosen by experience, such as the joint maximal and minimum angles $\theta_{max}, \theta_{min}$ and the initial phase offset ϕ_0. For each learned gait (pronking, bounding, walking), here only walking was learned, these parameters were chosen and manually tuned. The policy actions were first removed from the control, i.e. fixed gait frequency, and $\tilde{A} = 0, \tilde{\mathbf{q}} = \mathbf{0}$. That way, it was possible to ensure that the reference trajectory, describing a rough sketch of the desired gait, was good enough. Having a good reference trajectory was found to be crucial in making the PMTG work. Otherwise, the agent would first attempt to unlearn the reference, but since the action space was bounded, it was often unsuccessful in doing so, resulting in poor locomotion skills.

The reference trajectory Equation 2.2 was chosen to be periodic as a walking gait is periodic. Such a form made it possible to identify learnable parameters.

Only the joint residual was learned for the hip, i.e. for the hip joint angles $\theta_{tg}(t) = 0, \forall t$, which was found to be sufficient, and enabled to reduce the action space. The bigger the action space the less robust the resulting policy can be [31]. It is then recommended to

factor the action space as much as possible, as pointed out in [31].

The observation space was comprised of the height of the Center of mass (CoM), the first column of the rotation matrix defined from the euler angles of the base wrt. the world frame, the angular and linear velocity of the floating base with respect to the world frame, expressed in the body frame (or inertial frame) and at the CoM, the joint position and velocities, the leg phase $(cos(2\pi ft), sin(2\pi ft))^T$, the body linear velocity expressed in the world frame, the previous action a_{t-1}, and the joint position targets of the previous two control time steps.

The policy was modeled as a Multi-layer Perceptron (MLP), as in [4] for instance and as is commonly used in RL applications, of two hidden layers of dimensions $(256, 128)$, and $Tanh$ activations were chosen in the hidden layers and the output layer to promote smoother actions, smoother joint position and finally smoother locomotion. Using leaky $ReLU$ activations (defined in Equation 3.6) for the hidden layers led to high frequency and jerky motions. Such an architecture is used in [30].

The reward is a sum of different terms that provide a signal on the learned gait:

$$\begin{aligned} r_t =& \alpha_1 \min(v^x_{max}, v_x) - \alpha_2 \|Q - Q_0\|_2^2 - \alpha_3 \|\tau\|_2^2 - \alpha_4 \|\dot{\mathbf{q}}\|_2^2 \\ &+ \alpha_5 v^x_{max} \exp\left(-\frac{1}{2}\left(\frac{v_x - \bar{v}_x}{v^x_{max}}\right)^2\right) - \alpha_6 \frac{|v_x - \bar{v}_x|}{v^x_{max}} \\ &- \alpha_7 |v_y| - \alpha_8 |v_z| - \alpha_9 \|\mathbf{q}_\pi(t) - 2\mathbf{q}_\pi(t-1) + \mathbf{q}_\pi(t-2)\|_2^2 \end{aligned} \tag{2.3}$$

The reward coefficients α_i are all chosen to be positive. The reward coefficients/hyperparameters were experimentally found to lead to an optimal policy:

$$\boldsymbol{\alpha} = (0.1, 1.0, 10^{-5}, 10^{-4}, 0.4, 0.4, 1.0, 1.0, 10.0)^T$$

v denotes the body velocity in the world frame expressed at the CoM. v_x is the current velocity of the robot along the x axis of the world frame, v^x_{max} is the maximal allowed velocity. $Q \in \mathbb{R}^4$ denotes the body orientation in quaternions, and $Q_0 = (1, 0, 0, 0)^T$ is the

initial body orientation corresponding to a torso aligned with the world x axis having no roll angle. τ are the motor torques produced from the joint angle commands or targets with a Position-Derivative controller. Each term has its own use:

1. Prevents the robot from going too fast.
2. Incentivizes the policy from staying upright, i.e. not fall and limit the pitch and roll angles of the base.
3. Limits the joint torques. Policies producing actions resulting in high torques are less likely to be transferred. They produce noisier walking behaviors (harsher contacts) and consume more energy.
4. Limits the joint velocities, promoting low frequency and smoother motions.
5. Rewards the agent for tracking the desired or target body velocity $\bar{v}_x$.
6. Penalizes the agent for any divergence from the body velocity tracking, which will make the agent track the target body velocity faster than when using only term (5).
7. Limits lateral velocity to promote straight-line walking along the x axis.
8. Limits vertical velocity to promote straight-line walking along the x axis.
9. Promotes smoother joint targets by limiting the first derivatives of the joint targets.

The agent was trained using the on-policy RL algorithm Proximal Policy Optimization (PPO) [28], and the simulations were performed using the simulator *Raisim* [29].

Adding front pushes to the floating base was found to be crucial in learning a smooth and robust locomotion gait. It forced the agent to move its legs to move forward, and have bigger strides that are more natural for a walking gait. Experimental data comparing policies trained with different terrain friction coefficients, i.e. uniformly sampled for each environment in $[0.02; 2.0]$, and policies trained with front pushes, i.e. front forces applied

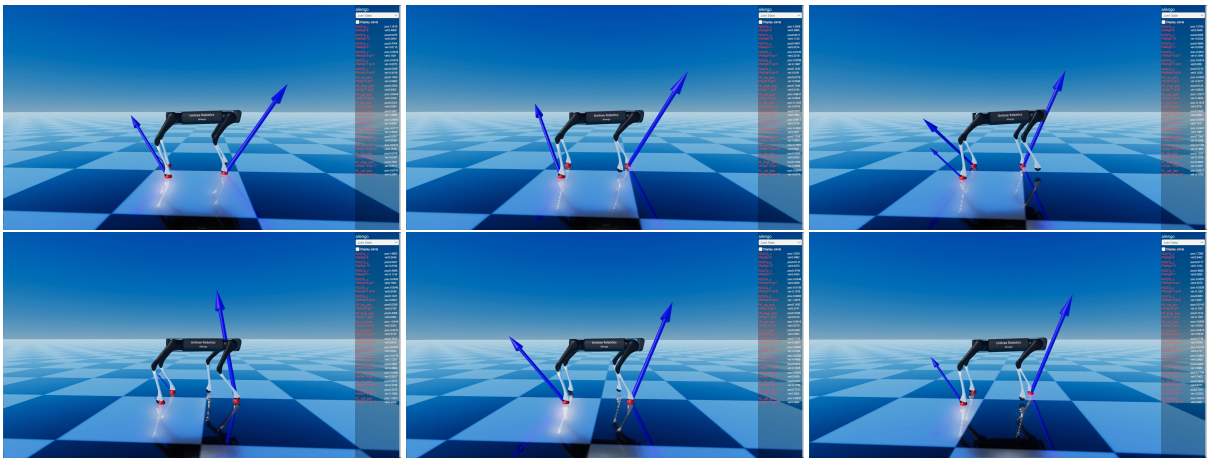

Figure 2.1: Rear foot slipping of the policy without front pushes.

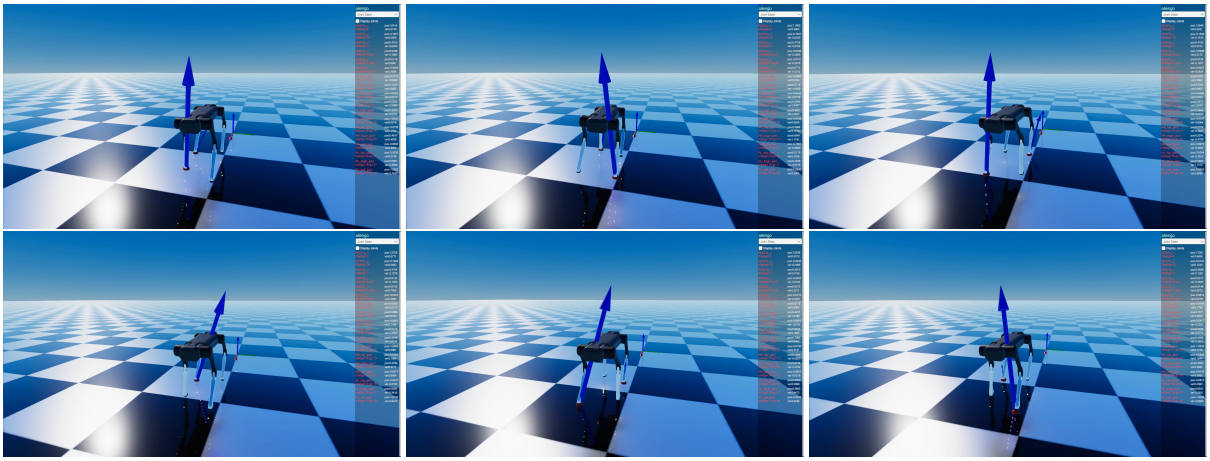

Figure 2.2: Front pushes during training removed the rear foot slipping/sliding as a slipping/sliding foot does not generate enough ground reaction forces to make the robot move forward against the front pushes.

in the opposite direction of the agent base motion and projected in the (x,y) plane and with norms uniformly sampled in $[0N; 60N]$ are available at this link. The videos show that without front pushes or friction, the policy slides rear feet. Friction produced a slower policy with smaller foot strides.

2.2.2 Results

The technical details presented in the previous section made it possible to obtain a reference-based walking policy. The best policy is presented in simulation in this video, please click

here to view.

2.3 Reference-free policy learning

2.3.1 Technical details

To avoid re-tuning the reference trajectory for different robots and morphologies, and to avoid having to define different action bounds and reward coefficients (for the family curve of the PMTG) for different locomotion skills, a reference-free algorithm based on [2] was implemented. The main idea here was to clip the joint actions, whose bounds were chosen to be for the hip $0.15rad$, and for the thigh and calf $0.4rad$, and to use a very general reward that would work across different morphologies. The bound of the hip is chosen smaller here as in the case of locomotion, the hip joint angle does not exceed $0.15rad$.

The actions are joint position residuals around a fixed nominal joint position. The policy learns both the mean and the standard deviation of the joint position residuals, hence $card(\mathcal{A}) = 24$. The standard deviation is only used for training as it promotes exploration (a minimum standard deviation is fixed). The actions are bounded around the nominal joint configuration to limit the space to explore and prevent some undesirable behaviors. Actions are also filtered using an average filter of window length 5. The average filter does an average of the 5 previous actions after bounding them individually. The average filter is defined as:

$$a_t = \frac{\sum_{i=0}^{N_w-1} a_{t-i}}{N_w} \tag{2.4}$$

where N_w is the window length of the average filter, and $a_t \in \mathbb{R}^{12}$ is the vector of joint positions at step t.

The observations are comprised of the joint positions and velocities, the roll and pitch angles of the floating base, the body/floating base angular velocity, the relative foot positions with respect to the body expressed in the body frame, and the previous $(t-1)$ roll, pitch, and angular velocity of the base, and relative foot positions.

The reward is a summation of three different terms as presented in [2]:

1. Staying upright, balance: $\alpha_1 \left(1 - \Omega(\sqrt{\phi(t)^2 + \theta(t)^2}, w_1)\right)$, with $w_1 = \frac{\text{atanh}(\sqrt{0.95})}{0.4}$, ϕ, θ respectively roll and pitch angles.

2. Forward walking (in the body x axis direction): $\alpha_2 \Gamma(v_x)$

3. Moving the legs in the direction the robot walks: $\alpha_3 \Gamma(\frac{1}{N} \sum_{i=1}^{N} v_{\text{swing}}^{x,i})$. N is the number of legs. $v_{\text{swing}}^{x,i}$ is the swing velocity (velocity in the aerial phase) of the foot i projected along the x axis of the body frame.

Where the saturation function $\Omega(x, w) = \tanh |xw|^2$ maps the input $x \in \mathbb{R}$ to $[0; 1]$ with the sensitivity parameter w. It can be seen as $|x|$ when $x \to 0$. $\Gamma(x) = \min((1 - \Omega(\omega_z, a))x, x)$, with $a = \frac{\text{atanh}(\sqrt{0.95})}{0.5}$ prevents the agent from being rewarded in the case where it moves forward (body or legs) but has a high yaw rate. This promotes the learning of a more stable locomotion gait that would follow the provided direction. For the experiments, the same reward coefficients as in [2] were used, that is $\boldsymbol{\alpha} = (0.1, 1, 0.5)$.

Finally,

$$r_t = \alpha_1 \left(1 - \Omega(\sqrt{\phi(t)^2 + \theta(t)^2}, w_1)\right) + \alpha_2 \Gamma(v_x) + \alpha_3 \Gamma(\frac{1}{N} \sum_{i=1}^{N} v_{\text{swing}}^{x,i}) \tag{2.5}$$

The architecture of the policy is an MLP of configuration (LayerNorm 256, 256, ELU, 100, ELU, 100, ELU, 24, Tanh). The standard deviation of the actions is bounded in $[0.3; 1.0]$ and the mean evolves in $[-1; 1]$. The LayerNorm layer learns to normalize its input from the training data, and,

$$\text{ELU}(x) = \begin{cases} x, & \text{if } x > 0 \\ \alpha * (\exp(x) - 1), & \text{if } x \leq 0 \end{cases}$$

where $\alpha = 1$.

Although [2] uses SAC (an off-policy RL algorithm) to train their agents, PPO was used here.

Using the same hyperparameters (reward coefficients and PPO training parameters) as in [2], which they claim worked across different morphologies, it was possible to obtain very natural and robust locomotion gaits as presented in subsection 2.3.3 and subsection 2.3.2, smoother (more natural) than the ones learned with the PMTG and presented in subsection 2.2.2. The reference-free policy has higher footsteps and uses the hip joint for balance. As presented in Figure 2.2 and subsection 2.2.2, the walking gaits generated by the PMTG look more "robotic". The different robots trained in simulation were A1, Aliengo, Anymal, Anymal C, and Laikago. Yet, these results (subsection 2.3.3 and subsection 2.3.2) were very hard to reproduce across different random seeds. Being able to reproduce an experiment across different random seeds is crucial in RL, as RL algorithms use stochastic gradient steps, and noise is often used in training to model sensor noise or avoid overfitting (i.e. data augmenation). Fixing the random seed of a program ensures that sampling from any distribution, in the same way, will give the same values. [2] uses 3 different random seeds, but does not tackle at all these issues of reproducibility. The agent would often learn a bounding or pronking gait (robot jumping on the front or rear feet) and not a walking gait depending on the random seed.

How to promote the learning of walking gaits? In order to make the learned gait more robust, and help the robot escape the local minima corresponding to bounding or pronking gaits, entropy was used. PPO is an entropy-based RL algorithm which means that it uses in the objective an entropy term to maximize exploration [32]. It was found to increase the reproducibility across different random seeds, but did not prevent the agent from learning less robust gaits. An entropy of 0.01 was found to produce the best reproducibility, i.e. policies trained with this value were more likely to produce walking gaits than other less robust gaits. An Entropy value too high makes the learning too unstable, and a value that

is too low makes the policy more likely to learn a pronking or bounding gait. This value was also found to produce the best comprise between exploration and unstable learning as in [28, 33]. From [28] the entropy term can be defined as:

$$H_t = -\beta \sum_{a_t \in \mathcal{A}} \pi(a_t|o_t) log(\pi(a_t|o_t)) \tag{2.6}$$

where β denotes the entropy coefficient and here $\beta = 0.01$, $\mathcal{A}$ is the action space. As the policy is a probability distribution this quantity tries to maximize the number of different produced actions given the current observations o_t. This quantity is an entropy bonus as it is always positive.

Even though the entropy made the agent explore more states during training, and thus made the agent more likely to escape bad initial local minima [33], it was not enough to learn a walking gait consistently. As was also working for the PMTG, front pushes were used to promote the emergence of walking gaits. Front pushes along the x-axis of the inertial frame of the agent were sampled in $[0N; 20N]$ during training.

The learned policies were deployed on the A1 robot from unitree without falls. A common practice is to randomize the environment and the robot model, this is called *Domain Randomization* (DR) [34, 3], sometimes referred to as *Dynamics Randomization* when the randomized parameters are part of the robot model. The quantities that are randomized here are the control latency (time between observation and when the robot is actuated), the gains, and the inertia (mainly the mass of the trunk). Before randomizing too much, the idea is to see how well the raw policy can perform on the hardware without any DR. [35] showed that it was possible to deploy policies without DR. Too much DR can harm the performance of the policy and produce very conservative policies. For instance, high contact forces can emerge.

Since A1 is not a real-time control system control times can vary. Every time a position command is sent to the DC motors, the time delay will change. The main problem when

deploying is to deal with this asynchronous part that arise from the communication between the high-level computer (Jetson Xavier), the low-level computer, and finally the DC motors. These latencies are hard to predict. Without DR, control times were synchronized on the robot by adding time artificially if there was less latency. Of course, sometimes the latency is too big and the synchronization did not work. The randomization of the control latency is then crucial to deployment. The idea is to train a policy with a likelihood of 0.3 to repeat the previous command a_t at time t. A real number is uniformly sampled in $p \in [0.0; 1.0]$ and if this number $p < 0.3$ then $a_t = a_{t-1}$. From the results obtained on the hardware and presented in subsubsection 2.3.3, it was found that synchronizing control times as much as possible and randomizing control times was the most important. Other quantities were randomized such as mass, friction, proportional gains, but it did not improve the deployed policy. The mass was sampled following a gaussian distribution centered around the robot's true mass and with a standard deviation of $0.05kg$ for each new environment, as well as the friction uniformly in $[0.04; 2.0]$, and the proportional gain in $[20.0; 50.0]$. The randomized parameters were tested in isolation and together without increasing the transferability successes.

2.3.2 Results: Policy in simulation on multiple robots

In order to prove the generality of the approach, which is a reference-free one, different policies were trained for different robots. Click here to download the videos. The robots that were tested are: A1, Aliengo, Anymal B, and Anymal C. The policies were able to learn smooth walking behaviors for each morphology. These policies are not randomized and were not deployed on hardware. Only deployment on A1 was tackled.

2.3.3 Results: A1 walking policy

This section presents the best policy that was deployed on hardware. Simulating control delays during training made it possible to deploy a policy on hardware without falls.

Simulation

Click here to visualize the policy in simulation. Figure 2.3 presents the evolution of the average per step reward of a policy trained with delay randomization. The policy was successfully deployed as presented in subsubsection 2.3.3.

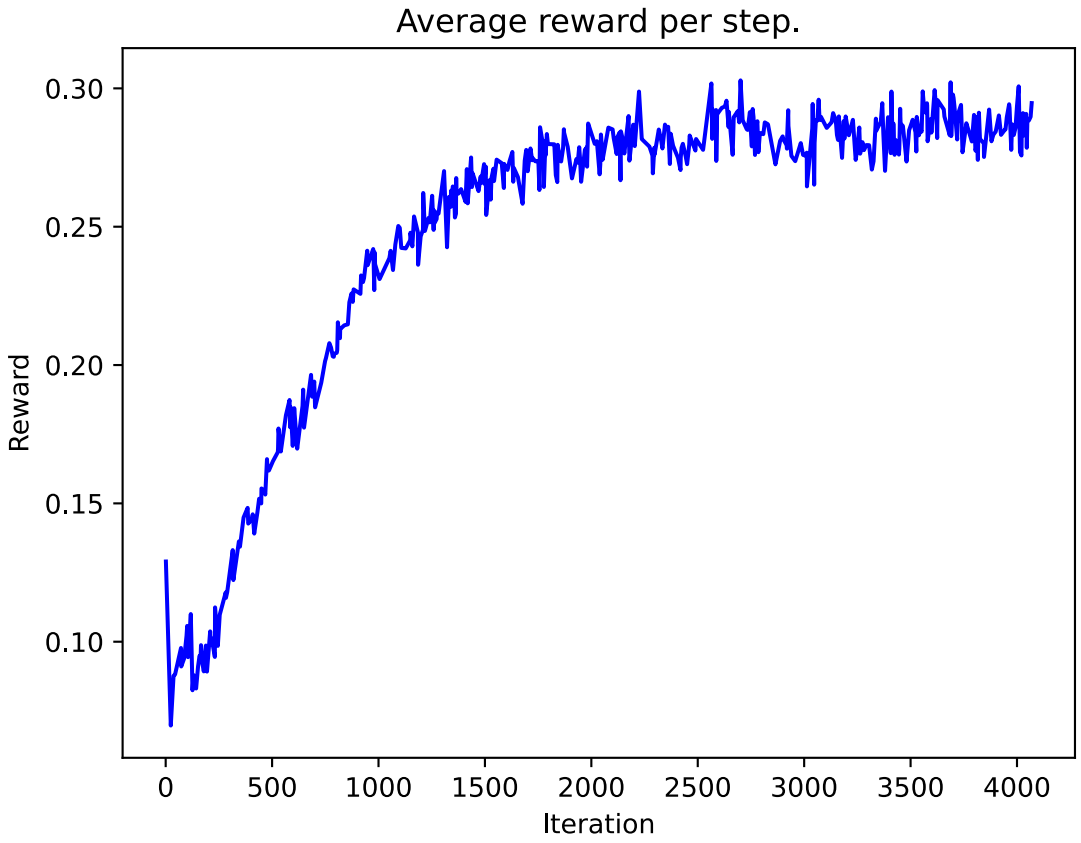

Figure 2.3: Average reward per control step for the training of the reference-free policy, which has been delay randomized and deployed on A1. The horizontal axis is the number of training iterations.

Hardware

The policy was deployed in different environments to evaluate its robustness and adaptability. Videos of the deployments can be downloaded at the following link. Performances between the 3 tested environments are similar, i.e., deployment without falls. The main is-

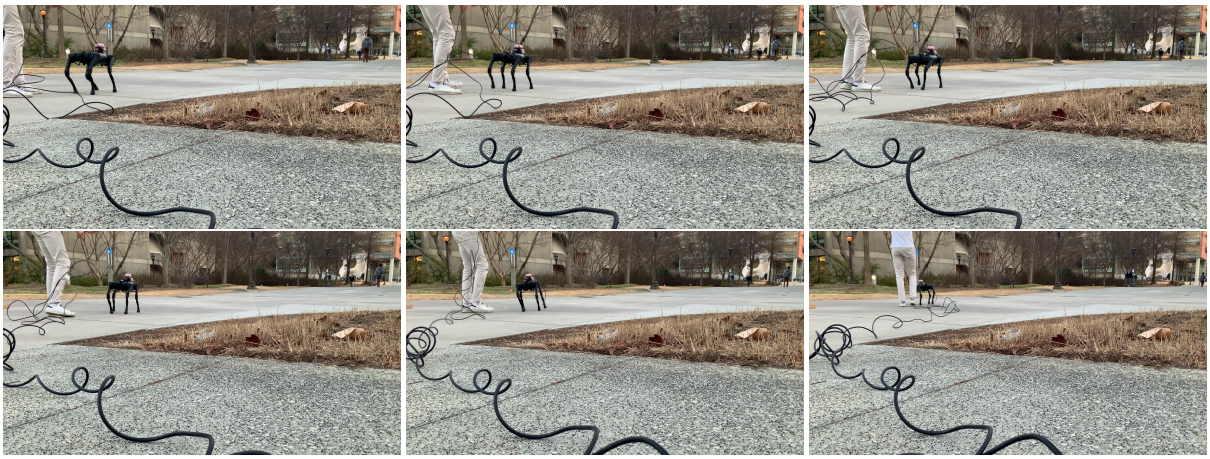

Figure 2.4: Hardware deployment of the walking policy.

sue with the deployed policy in all three environments is the drift to the right that appeared. The policy did not walk straight. It seems that this drift is only worse in the real world. Interestingly, the drift is bigger when the robot is tested on a hard floor than on a carpet.

This drift is not due to transfer as it is also present in the simulation. One thing that changed is the gains. In order to promote the most compliant and thus smooth and natural behaviors, the lowest possible gains are used in simulation as well as on hardware. In simulation $k_p = 25, k_d = 0.2$, and in the real-world $k_p = 40, k_d = 0.5$. Testing the policy in simulation with the real-world gains produces the same behaviors observed in the real world, the same drift. The test video can be downloaded at this link. Even training the policy with real-world gain did not solve this issue.

Work that tries to enforce symmetry in the simulation of the taken actions could be explored further as presented in [36], another idea would be to condition the policy based on the walking direction, making the walking direction an observable parameter of the learning problem.

Conclusion

Both the reference-based and reference-free approaches produced walking policies in simulation. The reference-free policy is smoother and more natural in simulation as shown

by the videos of the learned walking gaits. The reference-free policy has higher footsteps and uses the hip joint for balance. This is why only the reference-free was deployed on hardware. Except for the drift exhibited by the learned behavior, the learned reference-free policy was deployed on hardware without falls.

However, these techniques have been applied unsuccessfully to produce more expressive locomotion skills. These techniques can produce robust walking behaviors but lack generality for more complex locomotion skills. For instance, how is it possible to specify the transition between the walking and aerial phases in a jump? How is it possible to make it look natural? It seems that finding a way to specify the behavior is key. The next chapter tackles the generation of such skills through motion clip imitation.

CHAPTER 3
LEARNING EXPRESSIVE LOCOMOTION SKILLS THROUGH MOTION CLIPS

3.1 Introduction

Biological systems exhibit an infinity of locomotion skills and their variations. They can be creative in composing them together, and learn new skills. In short, they can be expressive. Even though recent approaches produce robust walking policies on real-world terrains [3, 37], and more diverse locomotion skills are tracked with individual policies mostly in simulation [38, 4, 39], they remain narrow in their scope. Rare are works that tackle generating expressive locomotion policies outside of a lab environment.

The papers, which deal with the latter problem, often limit their repertoire of motions (choose only similar motions, not acrobatic, static, or agile ones), use massive datasets of motion capture data (reaching thousands of motion clips for some skills), which are sorted or clustered before training to train different policies for each individual locomotion skills or motion clips, which exhibit similar kinematic trajectories. Most work do not even explore the composition of the learned locomotion skills and tackle the deployment of individual policies to the real world. Lastly, the state-of-the-art approaches, which learn to compose the learned skills, fragment the learning too much (training single policies for different skills), limiting the size of the locomotion skill repertoire, limiting the generalization capabilities, limiting the learning of natural and smooth transition between skills, making the deployment of the final policy harder (deployment is rarely tackled in that case), and limiting its evolution (adding new skills).

[5] is closest to the goal of this work. The latter paper uses 8 pre-trained experts based on reference-free reward shaping and composes the different skills with a gating neural

network, referred to as a Mixture of Experts in the literature [5, 4]. The learned locomotion skills are very close in nature, which makes the transitions easier to learn. One of our insights is that to achieve expressive locomotion as seen in biological systems, learning a vast repertoire of skills and their transitions is crucial. Transitions are very hard to specify through reward shaping techniques, motion capture data or just any motion data, which could be artificially generated, is much more suited to that task. Motion data is likewise crucial to specify skills as some more agile and acrobatic skills are intractable to reward specify. In that sense, this work looks at the problem through the lens of animation using motion capture data/motion data to specify behaviors and transitions as in [4]. Only a single policy is trained to learn the different skills. In the animation literature, motion capture data are often clustered beforehand to train a policy on each motion capture data cluster (clusters can different skills). As expressiveness requires a variety of motion skills and thus varied data, our approach prevents any manual preprocessing after the kinematic retargeting. Kinematic retargeting is the phase that adapts the motion data for the kinematic of the robot, which is trying to imitate the data as the data could be generated from another quadrupdal morphology). Motions are gathered in one single dataset.

Our second insight is that training information can be used to figure out a clustering, and based on this information, a training rule can be devised: *what can be trained together, can be clustered together.* Since the clustering is done before the actual training as in [38], it is solely based on kinematics, or manually designed criteria [4]. Using the training data it is possible to cluster motion clips based on kinematics and dynamics and to automate the clustering based on the agent progression. Different from a curriculum learning approach [40], focusing on mastering one task, this approach exploits learning different skills simultaneously. In addition, our approach develops the policy more iteratively. Usually, it is either from the lens of animation or from the lens of robotics. The first stage of the policy training focuses on learning to imitate locomotion skills, and learning to transition using privileged information based on Learning-By-Cheating [41]. The second stage focuses on

the deployment of a single policy.

In summary the main contributions are expected to be (not everything has been implemented at the time of the writing):

- Kinematically re-targeted motion capture dataset based on MANN [7].
- Single policy and dynamical controller, which can track a variety of locomotion skills based on a dynamic clustering approach.
- Real-world deployment based on Learning-By-Cheating [41].

3.2 Technical details

This work was realized in collaboration with Nitish Sontakke and Ren Liu. This thesis solely presents the contribution of this author.

3.2.1 Reference motions

The reference motion capture data are kinematic trajectories generated by the pre-trained NN provided with the code of [7]. By modulating the commands of the NN it is possible to generate a diverse set of locomotion skills: trotting, walking, turning while walking, and jumping. The targets are kinematically re-targeted to A1 for the tracking. The original reference trajectory sampling frequency is 60hz. The original trajectory is interpolated to be at 100hz for A1.

Other reference motions such as turning in place, standing on the four feet, standing on rear legs, laying, and squatting are manually generated.

3.2.2 Observation space

The observations comprise relative link positions and orientations with respect to the root joint of the robot, link linear and angular velocities as in [4]. For the A1 robot, the root joint is a frame with origin the center of mass of the floating base and oriented with x looking

towards the front legs and z looking upwards. [4] uses a phase variable in the observations to improve the tracking of periodic motions. Since our reference motions are not always periodic and the phase variable is similar to a frame identifier or index in the reference motion capture data, it was removed from the observations. Having a frame index in the observations did not show any improvements on the imitation learning performances in our case, and can cause the policy to link the frame with the action, which could lead to some form of overfitting. In that case, the policy would not be able to imitate the same motion capture data if the frames are shifted. The policy should link instead the robot state to the action because it is more successful for zero-shot generalization. Observations contain the generalized coordinates and velocities of the robot. These latter quantities are crucial to achieving good imitation performance. Based on [38] the future information of the reference trajectory is provided for the current target and 3 future steps at 20ms, 80ms and 1s in the future. For all these future steps the CoM and the end-effector positions, both expressed in the world frame, provide enough information on the reference for high-performance tracking.

The future reference information was shown to be critical to avoid drift from the reference. Tracking error tends to accumulate more and more without. Other quantities from the reference trajectory could have been given to the agent, such as the reference orientation of each link, the full generalized coordinates, the base orientation, and the joint positions for instance. End-effector positions and CoM position are the quantities from the reference that were crucial in limiting the drift, and since it is better to have a more compact observation space, only these quantities were given to the policy.

3.2.3 Reward

The reward is similar to [4]. It ensures tracking performance as well as motion quality. The imitation learning produces a dynamically feasible motion from a kinematically feasible motion. Four dense learning signals are used. The individual reward terms except the

penalty are bounded and positive.

An orientation term ensures that each rigid body of the robot tracks the rigid body of the reference motion.

$$r_{or} = \alpha_{or} \exp\left(\beta_{or} \sum_j (Q_j \ominus Q_{ref,j})^2\right) \tag{3.1}$$

where j iterates over all the rigid bodies of the robot, and Q denotes the quaternion representation, $\ominus$ the quaternion difference as defined in [4]. Q_j denotes the quaternion representation of the j-th body orientation wrt. the root joint as defined in [4].

An end-effector term ensures that the reference end-effector positions are tracked.

$$r_{ee} = \alpha_{ee} \exp\left(\beta_{ee} \sum_i \|\mathbf{p}_i - \mathbf{p}_{ref,i}\|^2\right) \tag{3.2}$$

where i iterates over the 4 end-effectors and $\mathbf{p}$ denotes the end-effector position w.r.t. the floating base and expressed in the world frame.

A CoM term ensures that the CoM of the reference is tracked. $\mathbf{c}$ denotes the CoM position.

$$r_{com} = \alpha_{com} \exp\left(\beta_{com} \|\mathbf{c} - \mathbf{c}_{ref}\|^2\right) \tag{3.3}$$

Lastly, an action penalty term ensures that the learned joint residuals stay as close as possible to the reference joint trajectory.

$$r_{pen} = \alpha_{pen} \|\mathbf{\Phi} - \mathbf{\Phi}_{ref}\|^2 \tag{3.4}$$

where $\mathbf{\Phi}$ denotes the joint positions.

For the policy tested in section 3.3 the reward coefficients producing the best tracking

results are:

$$\begin{aligned} \alpha_{or} &= 0.65, \beta_{or} = -2.0 \\ \alpha_{ee} &= 0.15, \beta_{ee} = -40.0 \\ \alpha_{com} &= 0.7, \beta_{com} = -12.5 \\ \alpha_{pen} &= -0.005 \end{aligned} \tag{3.5}$$

Contrarily to [4, 38] the reference velocity is not used for training. Neither in the reward and nor in the observations as it is not provided directly by the motion clips. Differentiating the joint positions to compute the joint velocities results in noise, as differentiation adds high frequencies, resulting in higher frequency actions and even a misleading signal for the policy, which resulted in unstable learning.

3.2.4 Action space

The policy outputs joint targets, which are then tracked using a PD controller ($k_p = 100$, $k_d = 4$). The actions are not absolute, but relative. The policy learns residuals w.r.t. the reference joint angles. It makes learning easier. The actions are bounded to make the problem even easier and make convergence faster. The bounds are $0.4rad$ for the hip joints and $0.6rad$ for the thigh and calf joints. The hip was limited a bit more since rare are motions that present huge hip angles. But the algorithm would also work with a bigger bound.

To avoid high frequencies in the actions, resulting from the agent exploration in the action space and also the high dimensionality of the observation space, the policy actions are filtered. An average filter of window length 2 was the best compromise between motions, which are too sluggish and too jerky.

3.2.5 Architecture

The architecture of the policy was kept as small as possible to promote generalization and reduce overfitting on the data, but was chosen big enough to be able to solve the given tasks.

The policy is composed of 2 layers: [256, 256]. Activations are *LeakyReLU* expressed as:

$$leakyReLU(\mathbf{x}) = max(\beta * \mathbf{x}, \mathbf{x}), x \in \mathbb{R} \tag{3.6}$$

where $\beta << 1$ is a positive constant.

3.2.6 Training

The RL algorithm used for training is an on-policy one, Proximal Policy Optimization (PPO) [32]. The framework when trained on individual motion types achieves incredible performance as presented in Appendix C. Appendix C shows the tracking performance of the trained policy on in-distribution (i.e. motion clips it was trained on). More experimental data is available this link, including videos.

Since we have a diverse set of motion capture clips, and among them some motions, which require self-body contacts such as standing on the rear legs, using the conventional body contact early termination is a dead-end. As in [4, 38] a reward-based termination is used. If the reward is not high enough then the episode is terminated. Giving a termination penalty to the agent was not necessary. Rather than using a termination criterion based on the entire reward as in, [4, 38], only the CoM reward term is used. It is also possible to define a threshold for the end-effector and orientation rewards. This choice was motivated by the fact that too much termination leads to local minima, and using one tracking quantity if possible enable the agent to explore more behaviors, and limits the manual thresholding that would occur by using the entire reward, i.e. the more thresholds there is to define, the more manual tuning there will be. Finding this threshold can be hard since it can vary from motion to motion, and if using a threshold for each reward term, the agent might be constrained too much from the start to learn anything (local minima). The thresholding could be automated, increasing the minimum performance to achieve gradually.

The training is much harder when a single policy is trained on a diverse set of motion

capture data. The issue is primarily the unbalanced dataset, i.e. some motion types are more represented, which will bias the policy toward learning an overfitted strategy for these types of motions. Instead of clustering the motion clips right away as in [38], the training performance is used as a clustering criterion. It was noticed that the policy would tend to learn some motions together at the start, but then overfit on these motions and never learn the others. In order to overcome that, an adaptive sampling technique is used. Different from a curriculum learning where it is a variation of one single task, here the agent is trained on all the locomotion tasks or locomotion types first (uniform sampling). Once some motions are well tracked, a bucket with these motions is formed and another with the unsuccessful motions. At each new episode each environment samples a motion clip to track. Drawing from the success bucket is only possible for a fraction of the environments, and other samples from the failure bucket. Once a motion is drawn in the failure bucket it cannot be placed back. This leads the agent to remember the learned motions, while not overfitting them and injecting novelty.

Different criteria can be defined to sort motions in these buckets. The early termination information is used here. If the motion clip cannot be tracked until the end it is a failure, else a success.

3.2.7 Dynamic clustering

What can be learned together, is similar in nature, and can be clustered together. Using the information of which motion clips are successfully tracked, motion clusters can be defined, and even used to bootstrap other policies.

3.3 Results

At the time of the writing, a single policy that could imitate diverse locomotion skills was obtained. The policy was trained on one single motion clip from different types of motions, that is jumping, trotting, walking, walking and turning, turning in place, rear-leg

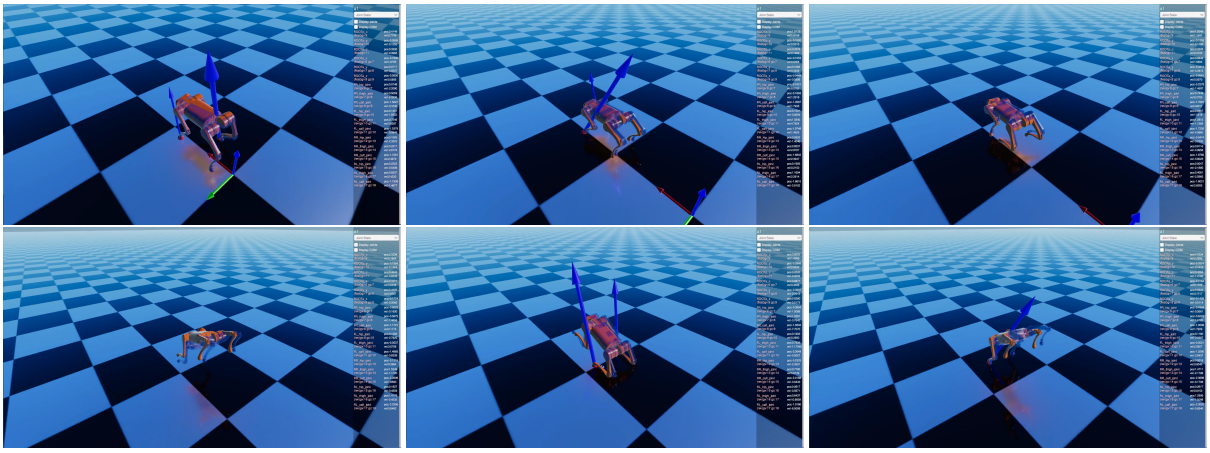

Figure 3.1: Key frames of the trained policy imitating a dynamic jump reference.

stand, crouching, sitting, stepping. Experimental data, videos of the tests of the policy in a simulation environment can be downloaded at this link. Figure 3.2 shows the training curve of the policy, i.e. the evolution of the total per-step reward throughout the training.

When tested in-distribution, i.e. on motion clips that the policy was trained on, the policy was able to track all the motion clips entirely. Appendix C presents the tracking performance for the trained policy on a dynamic jump.

To make sure that the policy did not overfit the training data. The out-of-distribution performance was tested on all the available data, i.e 554 motion clips composed of walking, trotting, turning and walking, and jumping motions. The policy was not trained on these motions. Appendix D presents the results. Trained on only 10 motion clips the policy was able to generalize successfully (track the motion clip until the end with high quality) to 250 motion clips (almost half of the dataset).

The other half of the out-of-distribution dataset will be tracked successfully using an adaptive sampling scheme that will enable the use of more training data and avoid biased policies toward more represented motion types.

Figure 3.2: Average reward per control step for training a policy on one motion clip of different motion types. The horizontal axis is the number of training iterations. The maximum achievable reward is 1.5. The achieved reward produced an acceptable tacking performance as presented in Appendix C, i.e. the policy produces the desired transitions and motions with high footsteps and a stable torso.

3.4 Conclusion

Preliminary results are very promising. Once the adaptive sampling and the dynamic clustering are implemented this work will get closer to a possible contribution. As shown in Appendix D, the policy actions are smooth, and torques are within the acceptable range (cf. Figure C.6, Figure C.7, Figure C.4, Figure C.5), which is promising for deployment.

CONCLUSION

This master thesis focused on designing expressive locomotion controllers, which can produce a variety of locomotion skills, including acrobatic and dynamic motions. The long-term goal is to use that expressivity to solve more complex real-world tasks. The first chapter presents how to design a walking policy with a reference-based and reference-free approach, from reward signal only and no motion clips. Using these techniques to learn a walking policy is feasible but already complex, as reward signals have to be carefully designed and tuned in both cases, and for the reference-based technique, the parameters and expression of the reference trajectory have to be carefully defined. These approaches limit the exploration of more acrobatic and complex motions. The key is to use motion clips and adapt them to the kinematics of the robot (motion retargeting) and to teach an agent to track the kinematic reference. Reward shaping is crucial to incentivize the agent to track the reference, but the learning framework is much more general than the PMTG and reference-free approaches and can be used for any type of motion provided at least one motion clip. Motion clips contain the transitions between motions and characterize natural motions better than any other techniques of motion generation (reward shaping, PMTG), which is crucial to learning complex locomotion skills.

When considering walking gaits solely, the approach tracking motion clips and presented in chapter 3 generates much more natural-looking motions than approaches using a direct specification of the motion in the reward as presented in chapter 2. The approach based on motion specification through motion clips generates gaits with higher footsteps and fewer base oscillations, i.e. the agent learns to use the hip joint for balance.

Current work focuses on producing one single policy, which could imitate a diverse repertoire of locomotion skills and deploy the learned policy on hardware.

Appendices

APPENDIX A

SUPPLEMENTARY MATERIALS

Supplementary materials are available at this link for download. It comprises videos of simulation and hardware results for the designed policies. Code for the work is available on github, please click here.

APPENDIX B

GYM ENVIRONMENT

Raisim is the simulator and physics engine that was used for this work [29]. For a more exhaustive list of parameters used for the code, please refer to the configuration (yaml files) provided with each environment.

APPENDIX C

TRACKING PERFORMANCE FOR A DYNAMIC JUMP

This part of the appendix presents the performance of the policy in imitating a dynamic jump. The performances for other motions are similar and can be downloaded here. The generalized coordinates comprise the position in the world frame, the orientation in quaternion representation wrt. the reference frame, and the joint positions. The reference is represented in black dashed lines.

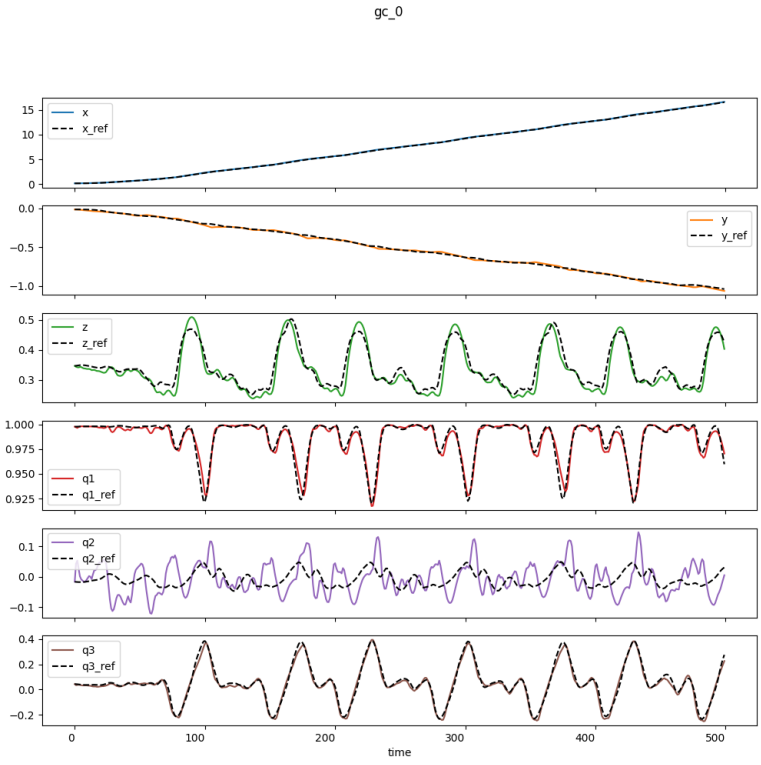

Figure C.1: Generalized coordinates.In order, the CoM position wrt . the world frame and orientation wrt . the world frame in quaternions, and the joint positions.

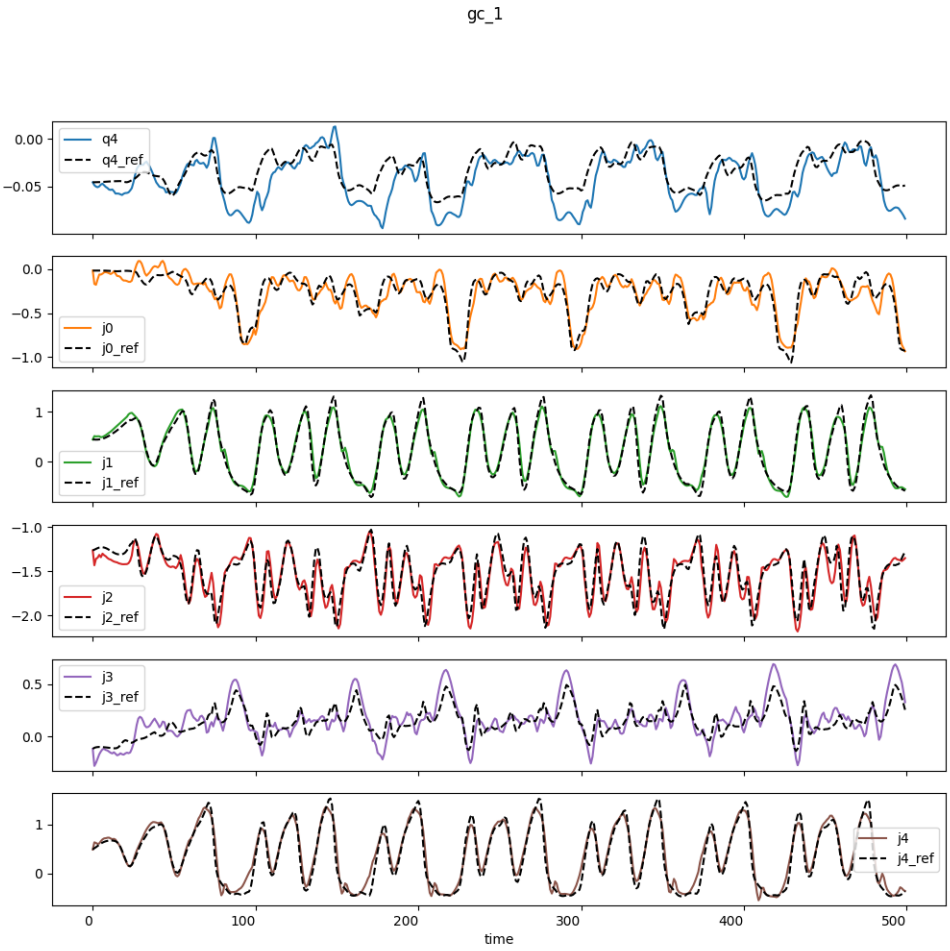

Figure C.2: Generalized coordinates. In order, the CoM position wrt . the world frame and orientation wrt . the world frame in quaternions, and the joint positions.

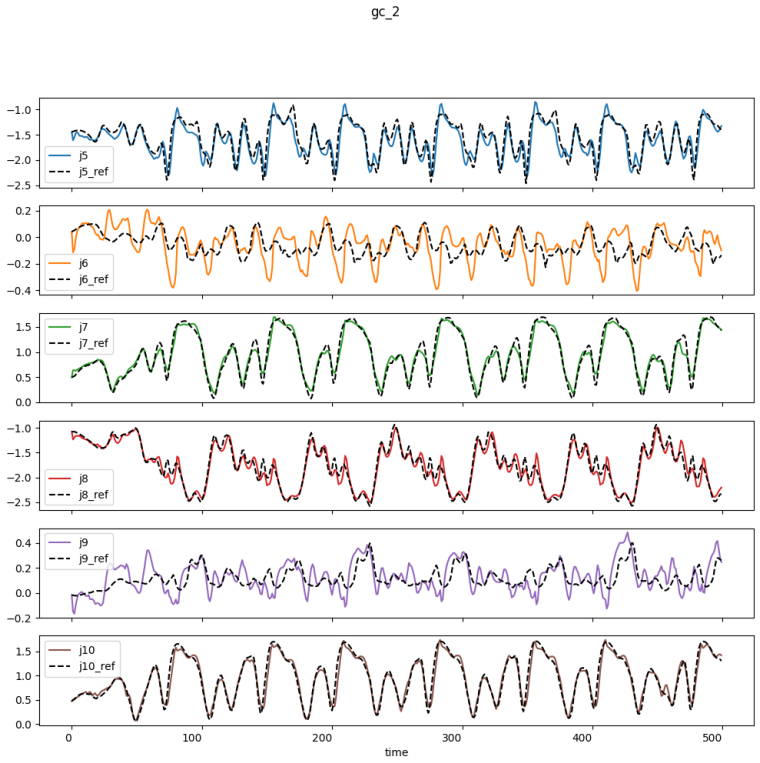

Figure C.3: Generalized coordinates.In order, the CoM position wrt . the world frame and orientation wrt . the world frame in quaternions, and the joint positions.

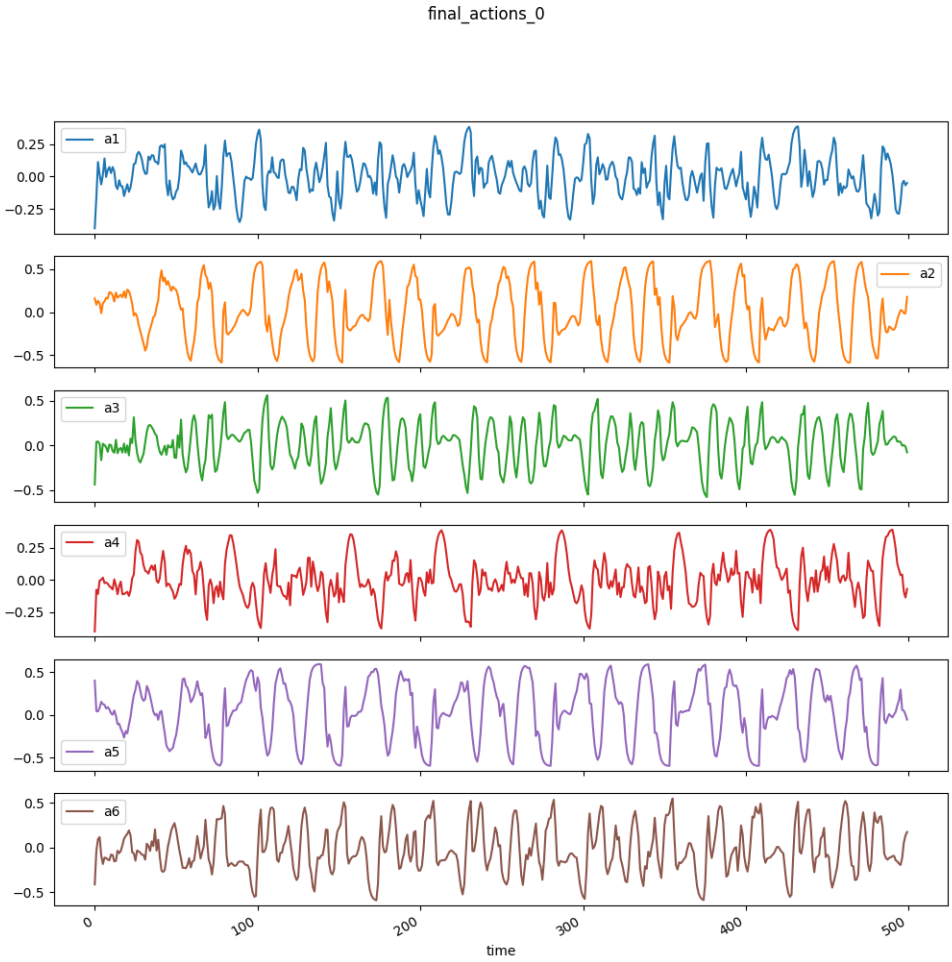

Figure C.4: Policy residual actions.

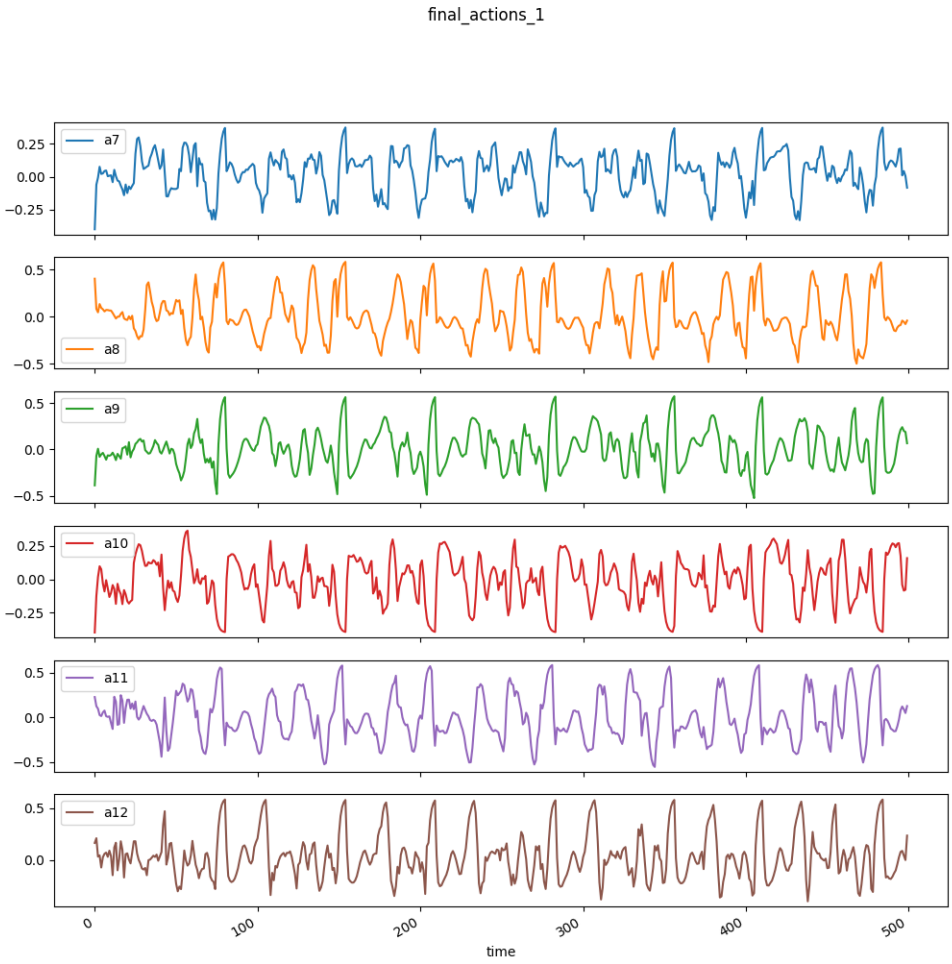

Figure C.5: Policy residual actions.

:

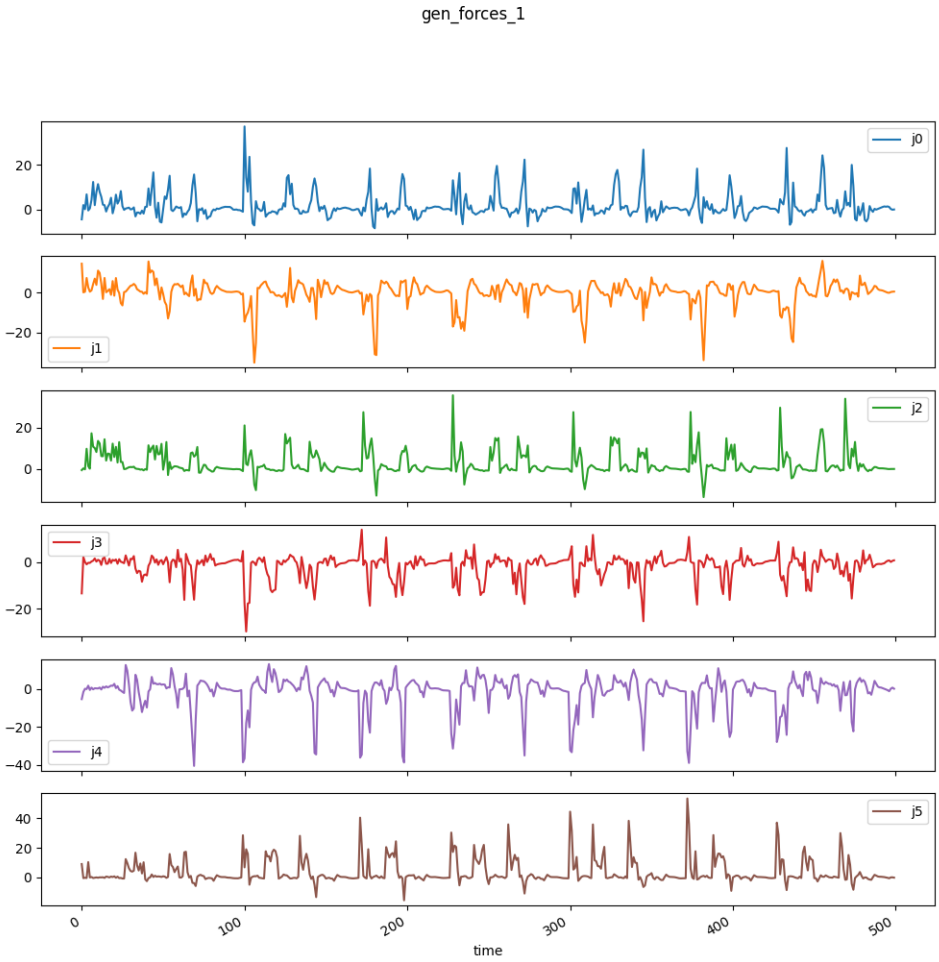

Figure C.6: Joint torques.

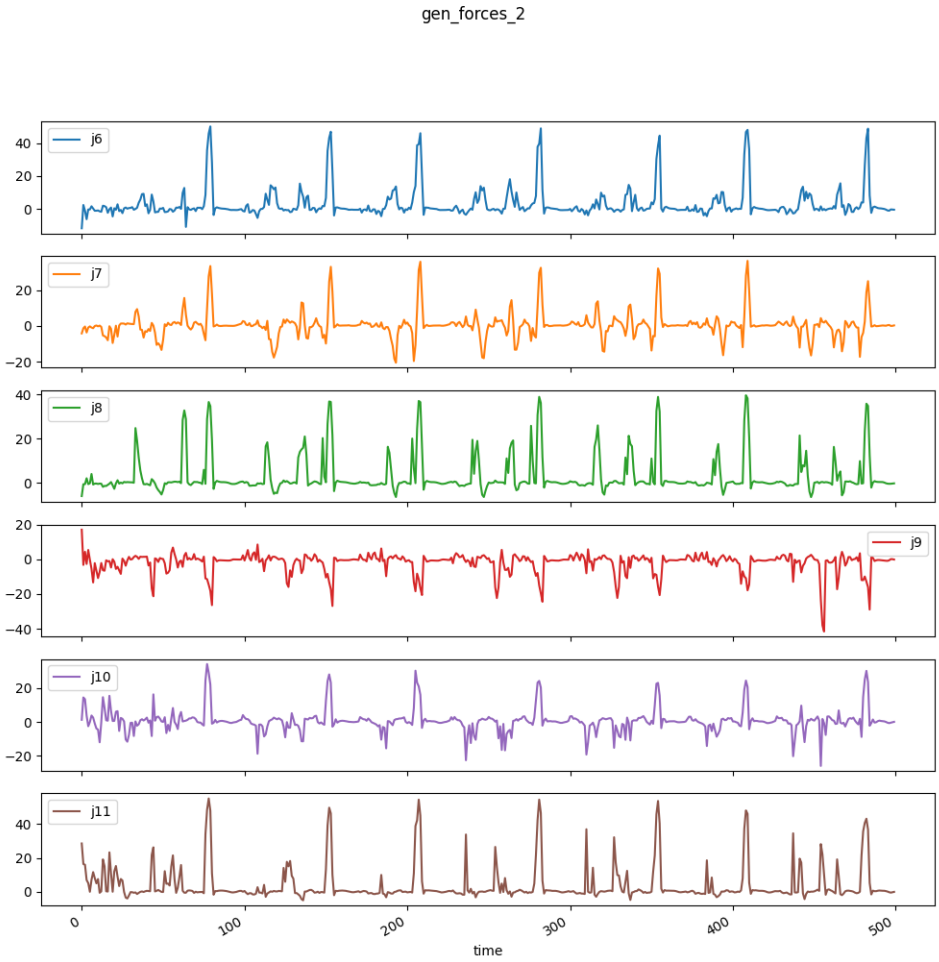

Figure C.7: Joint torques.

APPENDIX D

GENERALIZATION CAPABILITIES

This part of the appendix presents the out-of-distribution performance of a policy that was trained on one motion clip of each type, i.e. 10 motion clips. 554 motion clips were used for evaluation.

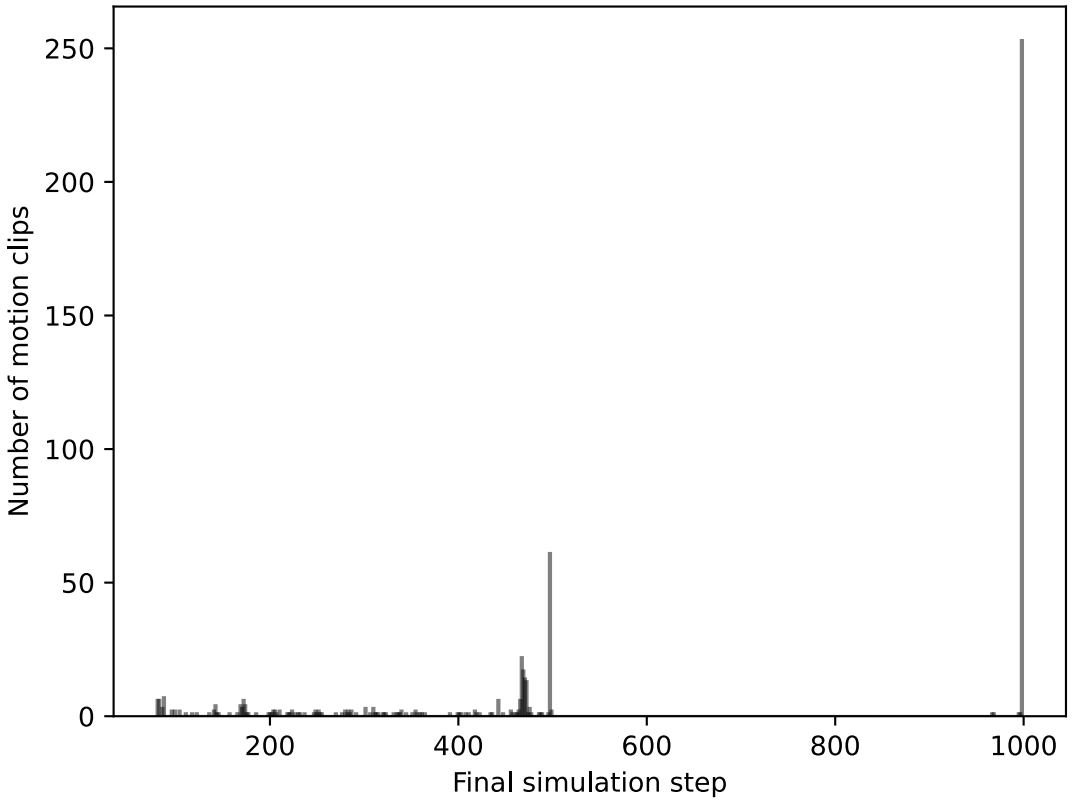

Figure D.1: Final simulation step reached in simulation. Histogram regrouping motion clips based on the final step reached in simulation (in a motion clip). The tested motion clips contained all 999 simulation steps. This plot shows that 250 motion clips, half of the dataset, were tracked until the end, and 70 more until half of the animation.

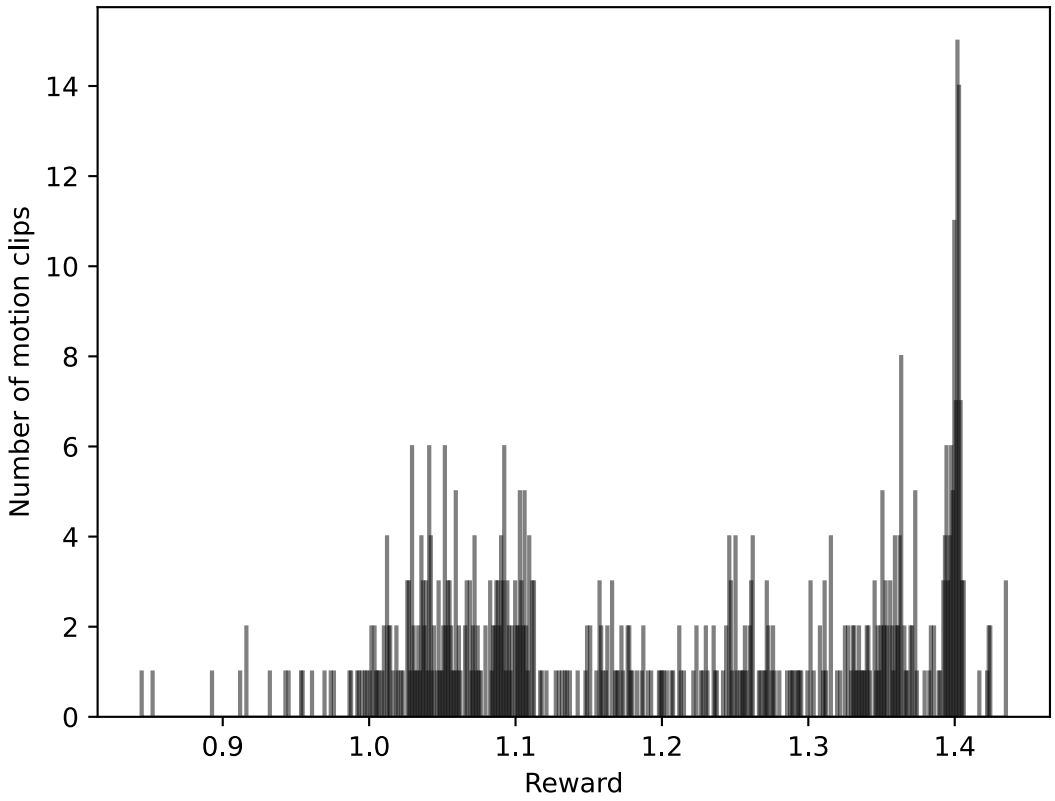

Figure D.2: Average reward per control step. Histogram regrouping motion clips based on the average reward per control step. This plot shows the average reward per control step, which is what the agent earns in one control step. The score is high for all motion clips, exemplifying that tracking is of high quality for the part of the reference motion that is tracked, even for out-of-distribution samples.

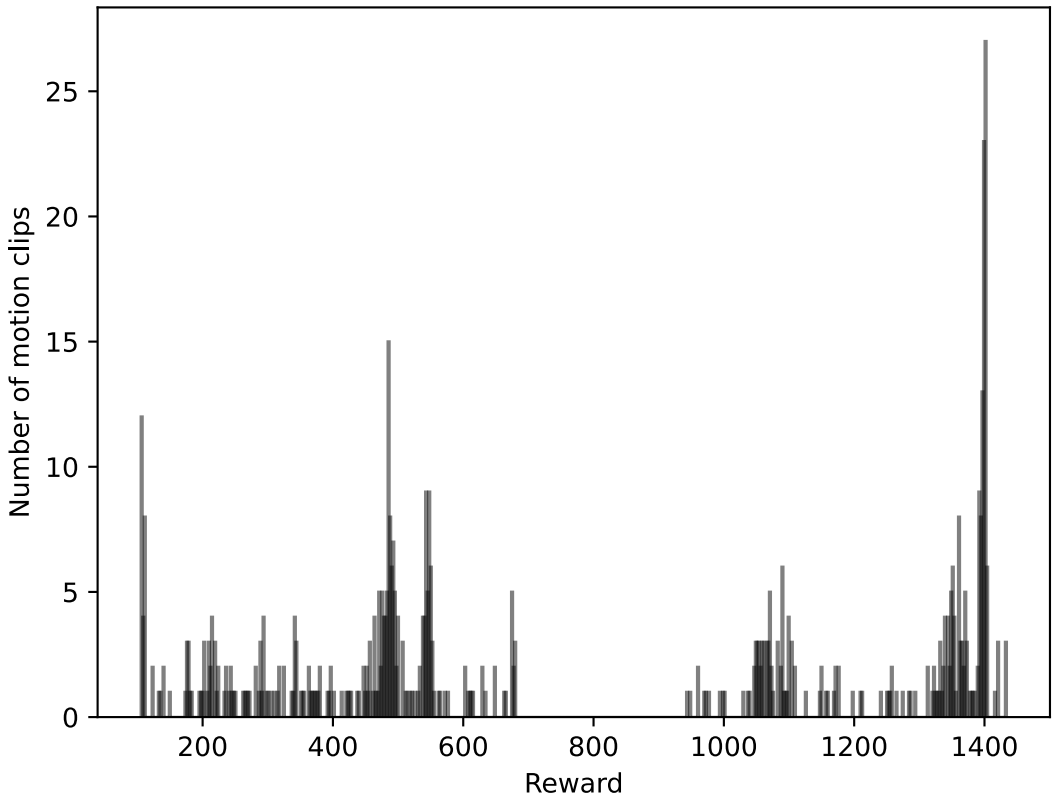

Figure D.3: Total reward for the entire episode or motion clip. Histogram regrouping motion clips with respect to their performance for the entire duration of the tracking even after termination. This plot is a combination of Figure D.2 and Figure D.1. Almost half of the out-of-distribution motions are successfully imitated.

REFERENCES

[1] G. Bledt, M. J. Powell, B. Katz, J. Di Carlo, P. M. Wensing, and S. Kim, "Mit cheetah 3: Design and control of a robust, dynamic quadruped robot," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2018, pp. 2245–2252.

[2] R. Hafner *et al.*, "Towards general and autonomous learning of core skills: A case study in locomotion," *CoRR*, vol. abs/2008.12228, 2020. arXiv: 2008.12228.

[3] T. Miki, J. Lee, J. Hwangbo, L. Wellhausen, V. Koltun, and M. Hutter, "Learning robust perceptive locomotion for quadrupedal robots in the wild," *Science Robotics*, vol. 7, no. 62, eabk2822, 2022. eprint: https://www.science.org/doi/pdf/10.1126/scirobotics.abk2822.

[4] X. B. Peng, P. Abbeel, S. Levine, and M. van de Panne, "Deepmimic: Example-guided deep reinforcement learning of physics-based character skills," *ACM Trans. Graph.*, vol. 37, no. 4, Jul. 2018.

[5] C. Yang, K. Yuan, Q. Zhu, W. Yu, and Z. Li, "Multi-expert learning of adaptive legged locomotion," *Science Robotics*, vol. 5, no. 49, eabb2174, 2020. eprint: https://www.science.org/doi/pdf/10.1126/scirobotics.abb2174.

[6] A. Iscen *et al.*, "Policies modulating trajectory generators," *CoRR*, vol. abs/1910.02812, 2019. arXiv: 1910.02812.

[7] H. Zhang, S. Starke, T. Komura, and J. Saito, "Mode-adaptive neural networks for quadruped motion control," *ACM Trans. Graph.*, vol. 37, no. 4, Jul. 2018.

[8] M. R. Reese, *The steam-powered pigeon of archytas – the flying machine of antiquity*, https://www.ancient-origins.net/history-famous-people/steam-powered-pigeon-002179, [Online; accessed Oct-2022], 2021.

[9] D. F. Hoyt and C. R. Taylor, *Gait and the energetics of locomotion in horses*, 1981.

[10] N. H. Hunt, J. Jinn, L. F. Jacobs, and R. J. Full, "Acrobatic squirrels learn to leap and land on tree branches without falling," *Science*, vol. 373, no. 6555, pp. 697–700, 2021. eprint: https://www.science.org/doi/pdf/10.1126/science.abe5753.

[11] *Agility robots*, www.agilityrobotics.com/robots, [Online; accessed 19-July-2021].

[12] *Unitree robots*, https://m.unitree.com/, [Online; accessed 19-July-2021].

[13] *Anybotics robots*, https://www.anybotics.com/anymal-autonomous-legged-robot/, [Online; accessed 19-July-2021].

[14] M. H. Raibert *et al.*, "Dynamically stable legged locomotion," *DSpace MIT*, 1985.

[15] A. W. 1. V. Winkler, *Optimization-based motion planning for legged robots*, Zurich, 2018-05-14.

[16] Z. Xie, X. Da, B. Babich, A. Garg, and M. van de Panne, "Glide: Generalizable quadrupedal locomotion in diverse environments with a centroidal model," *CoRR*, vol. abs/2104.09771, 2021.

[17] X. Da *et al.*, "Learning a contact-adaptive controller for robust, efficient legged locomotion," in *Proceedings of the 2020 Conference on Robot Learning*, J. Kober, F. Ramos, and C. Tomlin, Eds., ser. Proceedings of Machine Learning Research, vol. 155, PMLR, 16–18 Nov 2021, pp. 883–894.

[18] A. W. Winkler, C. D. Bellicoso, M. Hutter, and J. Buchli, "Gait and trajectory optimization for legged systems through phase-based end-effector parameterization," *IEEE Robotics and Automation Letters*, vol. 3, no. 3, pp. 1560–1567, 2018.

[19] J. Di Carlo, P. M. Wensing, B. Katz, G. Bledt, and S. Kim, "Dynamic locomotion in the mit cheetah 3 through convex model-predictive control," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2018, pp. 1–9.

[20] J. Koenemann *et al.*, "Whole-body model-predictive control applied to the hrp-2 humanoid," in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2015, pp. 3346–3351.

[21] A. Escande, N. Mansard, and P.-B. Wieber, "Hierarchical quadratic programming: Fast online humanoid-robot motion generation," *The International Journal of Robotics Research*, vol. 33, no. 7, pp. 1006–1028, 2014. eprint: https://doi.org/10.1177/0278364914521306.

[22] X. B. Peng, Z. Ma, P. Abbeel, S. Levine, and A. Kanazawa, "Amp: Adversarial motion priors for stylized physics-based character control," *ACM Transactions on Graphics (TOG)*, vol. 40, no. 4, pp. 1–20, 2021.

[23] X. B. Peng, A. Kanazawa, J. Malik, P. Abbeel, and S. Levine, "Sfv: Reinforcement learning of physical skills from videos," *ACM Transactions On Graphics (TOG)*, vol. 37, no. 6, pp. 1–14, 2018.

[24] J. Hwangbo *et al.*, "Learning agile and dynamic motor skills for legged robots," *Science Robotics*, vol. 4, no. 26, eaau5872, 2019. eprint: https://www.science.org/doi/pdf/10.1126/scirobotics.aau5872.

[25] J. Lee, J. Hwangbo, L. Wellhausen, V. Koltun, and M. Hutter, "Learning quadrupedal locomotion over challenging terrain," *Science Robotics*, vol. 5, no. 47, eabc5986, 2020. eprint: https://www.science.org/doi/pdf/10.1126/scirobotics.abc5986.

[26] J. Siekmann, K. Green, J. Warila, A. Fern, and J. Hurst, "Blind bipedal stair traversal via sim-to-real reinforcement learning," *arXiv preprint arXiv:2105.08328*, 2021.

[27] X. Da *et al.*, "Learning a contact-adaptive controller for robust, efficient legged locomotion," *arXiv preprint arXiv:2009.10019*, 2020.

[28] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, *Proximal policy optimization algorithms*, 2017. arXiv: 1707.06347 [cs.LG].

[29] J. Hwangbo, J. Lee, and M. Hutter, "Per-contact iteration method for solving contact dynamics," *IEEE Robotics and Automation Letters*, vol. 3, no. 2, pp. 895–902, 2018.

[30] J. Lee, J. Hwangbo, L. Wellhausen, V. Koltun, and M. Hutter, "Learning quadrupedal locomotion over challenging terrain," *Science Robotics*, vol. 5, no. 47, eabc5986, Oct. 2020.

[31] J. Tan *et al.*, *Sim-to-real: Learning agile locomotion for quadruped robots*, 2018. arXiv: 1804.10332 [cs.RO].

[32] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *CoRR*, vol. abs/1707.06347, 2017. arXiv: 1707.06347.

[33] Z. Ahmed, N. L. Roux, M. Norouzi, and D. Schuurmans, "Understanding the impact of entropy on policy optimization," *CoRR*, vol. abs/1811.11214, 2018. arXiv: 1811.11214.

[34] X. B. Peng, M. Andrychowicz, W. Zaremba, and P. Abbeel, "Sim-to-real transfer of robotic control with dynamics randomization," *2018 IEEE International Conference on Robotics and Automation (ICRA)*, May 2018.

[35] Z. Xie, X. Da, M. van de Panne, B. Babich, and A. Garg, "Dynamics randomization revisited: A case study for quadrupedal locomotion," *CoRR*, vol. abs/2011.02404, 2020. arXiv: 2011.02404.

[36] W. Yu, G. Turk, and C. K. Liu, "Learning symmetry and low-energy locomotion," *CoRR*, vol. abs/1801.08093, 2018. arXiv: 1801.08093.

[37] G. Ji, J. Mun, H. Kim, and J. Hwangbo, "Concurrent training of a control policy and a state estimator for dynamic and robust legged locomotion," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 4630–4637, Apr. 2022.

[38] J. Won, D. Gopinath, and J. Hodgins, "A scalable approach to control diverse behaviors for physically simulated characters," *ACM Trans. Graph.*, vol. 39, no. 4, Jul. 2020.

[39] X. B. Peng, Z. Ma, P. Abbeel, S. Levine, and A. Kanazawa, "AMP," *ACM Transactions on Graphics*, vol. 40, no. 4, pp. 1–20, Aug. 2021.

[40] Z. Xie, H. Y. Ling, N. H. Kim, and M. van de Panne, "ALLSTEPS: curriculum-driven learning of stepping stone skills," *CoRR*, vol. abs/2005.04323, 2020. arXiv: 2005.04323.

[41] D. Chen, B. Zhou, V. Koltun, and P. Krähenbühl, *Learning by cheating*, 2019.