

Robust Biped Locomotion Using Deep Reinforcement Learning on Top of an Analytical Control Approach

Mohammadreza Kasaei^a, Miguel Abreu^b, Nuno Lau^a, Artur Pereira^a, Luis Paulo Reis^b

^aIEETA / DETI University of Aveiro 3810-193 Aveiro, Portugal

^bUniversity of Porto, LIACC/FEUP, Artificial Intelligence and Computer Science Lab, Faculty of Engineering of the University of Porto, Portugal

Abstract

This paper proposes a modular framework to generate robust biped locomotion using a tight coupling between an analytical walking approach and deep reinforcement learning. This framework is composed of six main modules which are hierarchically connected to reduce the overall complexity and increase its flexibility. The core of this framework is a specific dynamics model which abstracts a humanoid's dynamics model into two masses for modeling upper and lower body. This dynamics model is used to design an adaptive reference trajectories planner and an optimal controller which are fully parametric. Furthermore, a learning framework is developed based on Genetic Algorithm (GA) and Proximal Policy Optimization (PPO) to find the optimum parameters and to learn how to improve the stability of the robot by moving the arms and changing its center of mass (COM) height. A set of simulations are performed to validate the performance of the framework using the official RoboCup 3D League simulation environment. The results validate the performance of the framework, not only in creating a fast and stable gait but also in learning to improve the upper body efficiency.

Keywords: Humanoid robots, modular walk engine, Linear-Quadratic-Gaussian (LQG), Genetic Algorithm (GA), Proximal Policy Optimization (PPO), Deep Reinforcement Learning (DRL).

Email address: Mohammadreza@ua.pt (Mohammadreza Kasaei)

1. Introduction

Developing a robust locomotion for bipedal robots is a challenging problem which has been investigated for decades. Although several walking approaches have been proposed and walking performance has considerably improved, it still falls short of expectations in certain domains, such as speed and stability. The question is *how is it that humans can constantly change their direction when running, while keeping their stability, but humanoid cannot?*

To find a good answer for this question, we start by reviewing recently proposed walking frameworks, consequently identifying four points of view related with the development of a fast and stable gait. In the first point of view, the fundamental framework's core is a dynamics model of the robot, based on which the walking planner and controller are designed. In this type of framework, to reduce the complexity of developing a whole body dynamics model, some constraints are considered. Based on these constraints, an abstract model is designed instead of a real whole body dynamics model [1, 2, 3, 4, 5, 6, 7, 8]. It should be mentioned that several studies exist where a whole body dynamics model is developed [9, 10, 11]. In the second point of view, the core of the framework is a set of signal generators which are coupled together to generate endogenously rhythmic signals [12, 13, 14, 15]. This type of framework is called Central Pattern Generator (CPG)-based framework and is inspired by the neurophysiological studies on invertebrate and vertebrate animals [16, 17, 18]. These studies showed that rhythmic locomotion like walking, running and swimming are generated by CPGs at the spinal cord that are connected together in a particular arrangement. In this type of framework, oscillators are assigned to each limb, typically to generate the setpoints (position, torque, etc.). Most humanoid robots have more than 20 Degrees of Freedom (DOF), therefore, adjusting the parameters of the oscillators is not only difficult but also trial-intensive [19]. Moreover, there is not a straight way to adapt sensory information to the oscillators. In the third point of view, walking trajectories are generated based on a heuristic algorithm such as reinforcement learning (RL), genetic algorithm (GA), etc [12, 20, 21]. In this type of framework, the walking trajectories will be generated after a training period which needs many sam-

ples and takes a considerable amount of time. During training, the framework tries to learn how to generate the walking trajectories, subject to an objective function. In the fourth point of view, the framework is designed by combining the aforementioned approaches [22, 23, 24, 25, 19]. This type of framework is generally known as a hybrid walking framework. It tries to leverage the different capabilities of each approach to improve the final performance.

After studying all types of humanoid walking frameworks, to find the answer for the question raised in the beginning of this section, let us look at how a baby starts to walk. It starts by learning to stand for a few seconds. It then improves the stability after many experiments, takes a few steps, learns how to maintain equilibrium while moving; until finally, after a long process of trial and error, a robust walking behavior emerges. This process shows how a human learns from previous experiences to improve its walking performance. Based on these explanations, we believe that the ability to learn from past experiences is the most important difference between human walking and robot walking. Particularly, a robot should be able to learn how to generate efficient locomotion according to different situations (e.g., learning to recover its balance from postural perturbations).

In the first two types of framework, the knowledge of robots is static, generally, and does not evolve from past experiences. Therefore, they need to at least re-tune the parameters to be able to adapt to new environments. In the third type of framework, the learning process typically does not consider any dynamics model and is designed based on learning from scratch, which is trial intensive and not applicable to a real robot directly. Several research groups have been exploring how to learn from previous experiences to improve stability and robustness. We believe the fourth type is the best approach to develop a robust biped locomotion framework.

In this paper, we propose a tight coupling between analytical control approaches and machine learning (ML) algorithms to develop a robust walking framework. Particularly, our contribution is a biped locomotion framework composed of two major components — an analytical planner and controller; and a fully connected neural network. The former is responsible for optimally controlling the overall state of the robot based on an abstract dynamics model. It is also responsible for generating reference

trajectories using dynamic planners with genetically optimized parameters and overcome uncertainties up to a certain degree. The latter component — a fully connected network — is optimized with reinforcement learning to control the arms residuals and the COM height of the robot, thus improving the upper body efficiency, which impacts the overall stability and speed of the robot.

The remainder of this paper is structured as follows: Section 2 provides an overview of related work. In Section 3, the concept of ZMP will be used to define a specific dynamics model which is composed of two masses. Afterwards, in Section 4, this dynamics model will be used to design an optimal controller which is able to track the walking reference trajectories, even in the presence of uncertainties. Section 5 explains how the problem of generating walking reference trajectories can be decomposed into five distinct planners. In Section 6, we will describe our learning approach and explain its structure. The overall architecture of the proposed framework will be presented in Section 7. In Section 8, three simulation scenarios will be designed to validate the performance of the proposed framework. According to the simulation results, its discussion and comparison with related work will be provided in Section 9. Finally, conclusions and future research are presented in Section 10.

2. Related Work

Several of the proposed walking frameworks are based on learning approaches to generate a stable locomotion for biped and multi-legged robots. Using ML algorithms for biped locomotion has made remarkable progress recently. These studies showed that using these algorithms on top of analytical approaches can improve robustness and performance significantly [22, 19]. In the remainder of this section, some recent proposed walking frameworks will be categorized and reviewed, focusing on those that use ML algorithms to improve their performance.

2.1. Combination of Model-based Walking and ML algorithms

MacAlpine et al. [22] designed and implemented a learning architecture to enable a humanoid soccer agent to perform omnidirectional walk. In their architecture, the

overall dynamics of a humanoid robot is abstracted by a double inverted pendulum model which is parameterized to be able to learn a set of parameters for different tasks. The performance of their framework has been validated using a set of simulations that have been designed using SimSpark ¹, a generic physical multiagent system simulator. The simulations results showed that their framework is able to learn multiple parameter sets according to the specified tasks.

Kasaei et al. [19] proposed a closed-loop model-based walking framework. Their dynamics model is composed of two masses that takes into account the lower and upper body dynamics of the robot. Based on this dynamics model, they generate walking reference trajectories and also designed an optimal controller to track these references. They showed the performance of their framework by performing a set of simulations using a simulated NAO robot in SimSpark. Moreover, they optimized the parameters using GA and showed that the maximum forward walking speed of the simulated robot reached 80.5 cm/s.

Carpentier et al. [26] proposed a generic and efficient walking pattern generator which is able to generate dynamically consistent motions. They argued that their approach is fast enough to generate the trajectory of COM along with the angular momentum according to the given configuration of contacts while the previous step is executing. Their method has been implemented on a real HRP-2 robot to demonstrate its interest. The experiment results showed that their method is able to generate long-step walking and climbing a staircase with handrail support.

Koryakovskiy et al. [27] proposed two approaches for combining a Nonlinear Model Predictive Control (NMPC) with reinforcement learning to compensate model-mismatch. The first approach deals with learning a policy to compensate control actions to minimize the same performance measure as their NMPC. The second approach was focused on learning a policy based on the difference of a transition predicted by NMPC and the actual transition. They performed a set of simulations to show the feasibility of both approaches and to compare their performances. The simulation results showed that the second approach was better than the first one. Moreover, They deployed the second ap-

¹<http://simspark.sourceforge.net/>

proach on a real humanoid robot named Robot Leo to perform squat motion to validate the performance of their approach.

2.2. Combination of CPG-based Walking and ML algorithms

Song et al. [28] designed CPG-Based Control walking framework which is able to generate stable walking, even on unknown sloped surfaces. In their framework, the walking patterns are generated based on CPG theory and a PI controller is designed according to gyroscope and accelerometer information, allowing the adjustment of the upper body's tilt angle to keep the robot's stability. They performed some experiments using a real NAO humanoid robot and the results showed that the robot is able to walk successfully on unknown slopes.

Missura et al. [29] proposed a walking framework which bootstraps a learning algorithm with a CPG-based walk engine. Their framework is composed of a feed-forward walking pattern generator, a state estimator and a balance controller. In their framework, while the robot is walking, the balance controller adjusts the step size based on the estimated error and also learns how to improve the walking performance by adjusting the swing leg parameters. The performance of their framework has been validated using a set of experiments on a real humanoid robot. The results showed that their framework is able to keep the robot's stability even after applying a severe push.

2.3. Combination of CPG-ZMP based walking and ML algorithms

Massah et al. [30] developed a hybrid CPG-ZMP controller to generate stable locomotion for humanoid robots. In their approach, a set of non-linear oscillators were used to generate walking trajectories and two controllers were developed to handle small and large disturbances. They optimized the walking parameters using the differential evolution (DE) algorithm. The performance of their approach was demonstrated in the Webots robot simulator using the NAO humanoid robot.

Liu et al. [31] proposed a CPG-ZMP based walking framework which is inspired by biomechanical studies on human walking. In their framework, walking reference trajectories are generated offline according to a point mass model. They used a PD controller to modify the reference walking patterns to keep the robot's stability. Moreover,

their framework takes the vertical motion of the upper body into account to generate almost stretched knees. The performance of their framework has been validated using a set of experiments on a real NAO humanoid robot. The results proved the improvement of walking stability and energy efficiency.

Kasaei et al. [25] developed a hybrid CPG-ZMP based walk engine for biped robots. Their walk engine has a hierarchical structure and it is fully parametric. They argued that this structure allows using a policy search algorithm to find the optimum walking parameters. To show this ability, they used an optimization technique based on Contextual Relative Entropy Policy Search with Covariance Matrix Adaptation (CREPS-CMA) [32] to tune the walking parameters. The performance of their walk engine has been validated by showing a fast and stable omnidirectional walk using a simulated Nao robot in Simspark (59cm/s).

2.4. Learning to walk from scratch

Abreu et al. [21] applied a reinforcement learning algorithm to develop a fast and stable running behavior from scratch. In their approach, the environment has been represented by 80 states and the action space is composed of 20 actions which were all the joints of a simulated humanoid robot. They used the Proximal Policy Optimization (PPO) based on the implementation provided by OpenAI [33]. The performance of their approach was shown by learning sprinting and stopping behaviors. The results demonstrated that both behaviors are stable and the sprinting speed stabilizes around 2.5m/s which was a considerable improvement.

Most of the aforementioned works combine a simplified model-based or a model-free approach with ML approaches to improve the performance of their walking. In the rest of this paper, we develop an optimal closed-loop walking pattern generator based on a more complex dynamics model which takes into account the vertical motion of the COM and the torso's dynamics. Besides, we use the PPO algorithm which is one of the most successful deep reinforcement learning methods, on top of our walking pattern generator to improve its robustness and efficiency and also to provide more human-like walking. An abstract overview of the proposed framework is depicted in Figure 1.

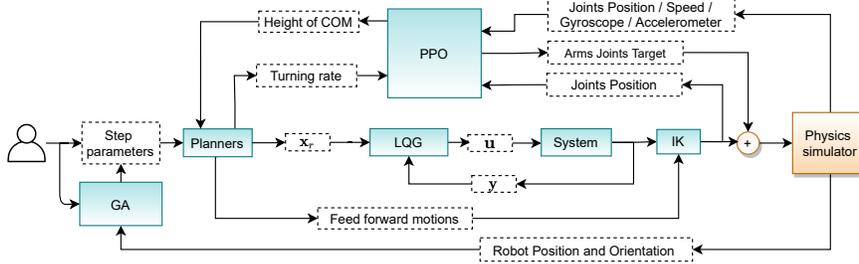


Figure 1: An abstract overview of the proposed framework. The highlighted boxes represent functional modules and the white boxes correspond to exchange data among them.

3. Dynamics Model and Stability Criteria

When designing a model-based walking, two general perspectives exist: (i) considering an abstract dynamics model which takes into account a trade-off between accuracy and simplicity; (ii) considering a whole-body dynamics model which is more accurate but, not only is it platform dependent but also resource-intensive due to its non-linear nature. In the rest of this section, the concept of Zero Momentum Point (ZMP) will be reviewed and then used to define an abstract dynamics model of a humanoid robot.

3.1. Zero Momentum Point

ZMP has been proposed in [34] and is currently one of the most successful metrics in the walking literature. Particularly, it is a point on the ground where the ground reaction force (GRF) acts to cancel the gravity and the inertia. Normal human walking is a periodic motion which can be decomposed into two main phases: (i) Single Support (SS) and (ii) Double Support (DS) [35]. During SS phase, only one foot is in contact with the ground and the other foot swings toward the next planned foot position. In this paper, we used the ZMP as our main criterion for analysing the stability of the robot while performing walking and it can be defined using the following equation:

$$p_x = \frac{\sum_{k=1}^n m_k x_k (\ddot{z}_k + g) - \sum_{k=1}^n m_k z_k \ddot{x}_k}{\sum_{k=1}^n m_k (\ddot{z}_k + g)}, \quad (1)$$

where n represents the number of parts that are considered in the dynamics model, m_k is the mass of each part, (x_k, \dot{x}_k) , are the horizontal position and acceleration, and (z_k, \ddot{z}_k) are the vertical position and acceleration of each mass, respectively.

3.2. Dynamics Model

Although considering a full body dynamics model is not impossible, it generally needs powerful computational resources. Therefore, it is not affordable for real-time implementation. To reduce the complexity of the model and its computation cost, the overall dynamics is approximated by an abstract model. Kajita and Tani [1] proposed an abstract model named Linear Inverted Pendulum Model (LIPM) which is a well-known abstract model in the community. LIPM is popular because it provides a simple, fast and efficient solution for walking dynamics that is suitable for real-time implementation. In this model, the overall dynamics of the robot is abstracted to a single mass that is connected to ground via a massless rod. Additionally, this model assumes that the vertical motion of the mass is restricted by a horizontally defined plane. According to these assumptions and using a set of predefined footsteps, the trajectory of Center of Mass (COM) can be obtained from a straightforward analytical solution which guarantees long-term stability. It should be mentioned that based on these assumptions, the equations in sagittal and frontal planes are equivalent and independent, therefore we just derive the equation in the sagittal plane. The schematic of this model is depicted in Figure 2(a). Using (1) and considering the LIPM's assumptions, the COM's motion equation can be obtained as follows:

$$\ddot{x}_c = \omega^2(x_c - p_x) \quad , \quad (2)$$

where $\omega = \sqrt{\frac{g}{z}}$ is the natural frequency of the pendulum, p_x and x_c represent the positions of ZMP and COM, respectively.

As aforementioned, LIPM tries to keep the COM's vertical position at a predefined position which causes the knee joints to be always bent. Indeed, walking with bent knees consumes more energy and does not resemble human walking [36]. To release this constraint and generate more energy efficient and human-like walking, a sinusoidal

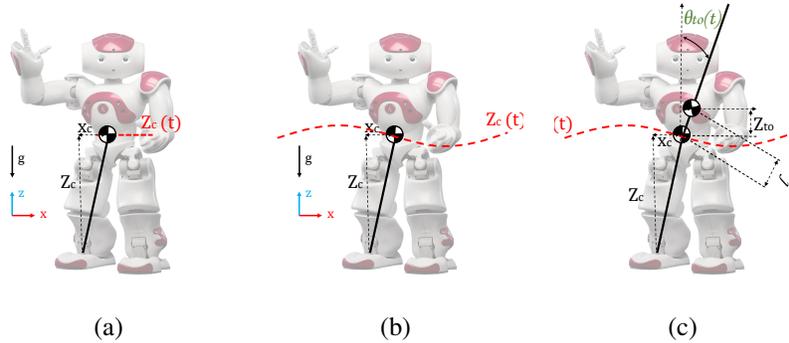


Figure 2: Schematics of the dynamics models: (a) LIPM; (b) LIPM with vertical motion of COM; (c) Proposed model.

motion is assigned to the vertical motion of the COM:

$$z_c = z_0 + A_z \cos\left(\frac{2\pi}{StepTime}t + \phi\right) \quad , \quad (3)$$

where z_0 denotes the COM's initial height, A_z is the amplitude, and ϕ represents the phase shift of the COM's vertical sinusoidal motion. The initial value of these parameters are determined by an expert. Additionally, a controller can be designed to adjust these parameters based on sensory feedback. Although the current version of the dynamics model is able to provide fast and stable walking, it is not good enough to generate a very fast walking. In fact, in some situations like when a push is applied, the COM accelerates forward, and, as a consequence, the ZMP goes behind the Center of Gravity (COG). In this situation, the robot tries to decelerate the COM by applying a compensating torque at its ankles, keeping the ZMP inside the support polygon. The compensating torque will be saturated once the ZMP is at the support polygon's boundary and, consequently, the robot is going to be unstable. In such situations, a human moves its torso to keep the ZMP inside the support polygon and prevent falling.

To consider the effect of the torso's motion in the dynamics model, another mass should be added to the dynamic model. This modification changes the dynamics model to be non-linear. Therefore, it does not have an analytical solution and it should be solved numerically. Biomechanical analysis of human walking showed that the torso motion can be represented by a sinusoidal function whose motion parameters are dependent on the current robot state, and terrain conditions. The interesting point is that

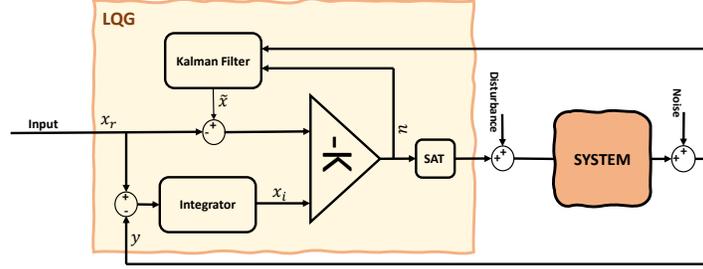


Figure 3: Overall architecture of the proposed controller.

if the torso is considered as a mass with a small sinusoidal movement relative to the hip ($\sin(\theta_{to}) = \theta_{to}$), the dynamics model can keep its linearity. The schematic of this model is depicted in Figure 2(c) and it can be represented by the following equation:

$$\ddot{x}_c = \mu(x_c + \frac{\alpha l}{1 + \alpha} \theta_{to} - p_x) - \frac{\alpha \beta l}{1 + \alpha \beta} \ddot{\theta}_{to} \quad (4)$$

$$\alpha = \frac{m_{to}}{m_c}, \quad \beta = \frac{z_{to}}{z_c}, \quad \mu = \frac{1 + \alpha}{1 + \alpha \beta} \omega^2, \quad x_{to} = x_c + l \theta_{to}$$

where x_{to} denotes the position of torso, θ_{to} , l are the angle of torso and length of torso, m_c, m_{to} represent the masses of lower body and torso, z_{to} , z_c are the torso and COM height, respectively.

4. Controller

In this section, an optimal controller will be designed for tracking the reference trajectories to minimize the tracking error. To do that, Equation (4) will be represented as a linear state space system. Then, this system will be discretized to be used in a discrete-time implementation. Afterwards, we will explain how this system can be used to design an optimal controller. The process of designing the controller starts by

defining a linear state space system based on Equation 4:

$$\begin{aligned}
 \underbrace{\frac{d}{dt} \begin{bmatrix} x_c \\ \dot{x}_c \\ \theta_{to} \\ \dot{\theta}_{to} \end{bmatrix}}_X &= \underbrace{\begin{bmatrix} 0 & 1 & 0 & 0 \\ \mu & 0 & \frac{\mu\alpha l}{1+\alpha} & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix}}_A \underbrace{\begin{bmatrix} x_c \\ \dot{x}_c \\ \theta_{to} \\ \dot{\theta}_{to} \end{bmatrix}}_X + \underbrace{\begin{bmatrix} 0 & 0 \\ -\mu & \frac{-\alpha\beta l}{1+\alpha\beta} \\ 0 & 0 \\ 0 & 1 \end{bmatrix}}_B \underbrace{\begin{bmatrix} p_x \\ \ddot{\theta}_{to} \end{bmatrix}}_u \\
 y &= \underbrace{\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}}_C \underbrace{\begin{bmatrix} x_c \\ \dot{x}_c \\ \theta_{to} \\ \dot{\theta}_{to} \end{bmatrix}}_X.
 \end{aligned} \tag{5}$$

The presented system is a continuous system and should be discretized for implementation in discrete time. To discretize this system, we assume that $\dot{x}_c, \dot{\theta}_{to}$ are linear. Therefore $p_x, \ddot{\theta}_{to}$ are constant within a control cycle. Thus, the discretized system can be represented as follows:

$$\begin{aligned}
 X(k+1) &= A_d X(k) + B_d u(k) \\
 y(k) &= C_d X(k)
 \end{aligned} \tag{6}$$

where k represents the current sampling instance, A_d, B_d, C_d are the discretized versions of the A, B, C matrices in (5), respectively.

According to this discretized dynamics model, an optimal closed-loop controller can be designed to track the reference trajectories. This controller is a Linear-Quadratic-Gaussian (LQG) which is composed of two main modules: a state estimator and an optimal controller gain. The overall architecture of this controller is depicted in Figure 3. In the remaining of this section, each module will be explained and the overall performance of the controller will be validated.

4.1. State Estimator

An LQG controller is able to track the reference trajectories even in the presence of measurement noise. This controller uses a state estimator to cancel the effect of uncertainties which can be raised because of many aspects like errors in modeling the

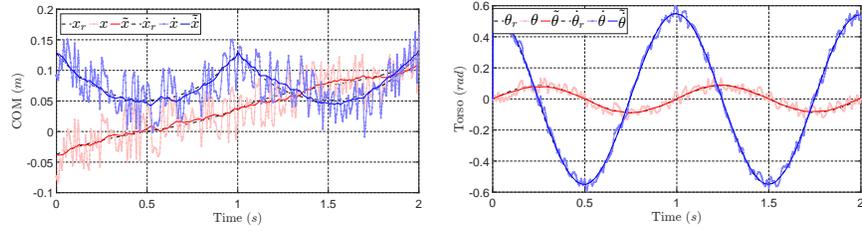


Figure 4: Simulation results of examining the state estimator performance. In this simulation, the measurements are affected by a Gaussian noise $\mathcal{N}(0, 6.25e-4)$ to simulate uncertainties. In these plots, light-blue and light-red lines represent the measurements, solid-blue and solid-red lines are the estimated values, dashed black lines represent the references.

system, sensor noise, backlash of the gears, etc. In particular, this controller uses a state estimator to estimate the current state of the system according to the control inputs and the observations.

In our target framework, we considered that the position of the joints is available through measurements and the torso orientation can be obtained based on an Inertial Measurement Unit (IMU) information which is mounted on the torso. Based on the joint information and using a Forward Kinematic (FK) model of the robot, the current configuration of the robot can be estimated. In this estimation, the support foot is considered to be in flat contact with the ground, which is not always true. Therefore, the whole body orientation with respect to the ground should be added to this estimation. To do so, the IMU information is used to rotate the current configuration. Based on this configuration, the COM position can be estimated at each control cycle and its velocity can be obtained from its position’s derivative, followed by a first-order lag filter.

To validate the performance of this module, a simulation has been designed. In this simulation, the observations are modeled as a stochastic process by applying additive Gaussian noise to the measured states. The simulation results are shown in Figure 4. According to the simulation results, the state estimator is able to estimate the states perfectly.

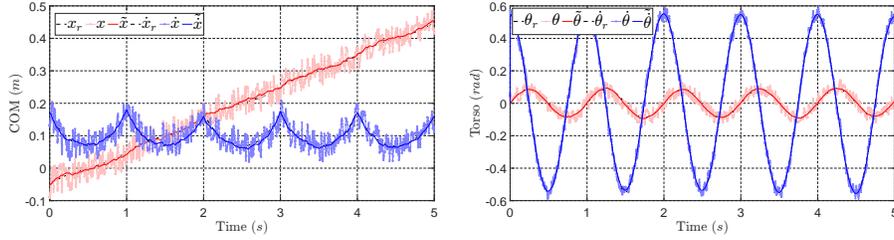


Figure 5: Simulation results of examining the controller performance in presence of noises. In this simulation, the controller should track a references trajectories in presence of the noise $\mathcal{N}(0, 6.25e-4)$.

4.2. Optimal Gain

As shown in Figure 3, the optimal control law is obtained using the following formulation:

$$u = -K \begin{bmatrix} \tilde{x} - x_r \\ x_i \end{bmatrix}, \quad (7)$$

where \tilde{x} , x_r denote the estimated states and the reference states, respectively. x_i is the integration of error which is used to eliminate the steady-state error, K represents the optimal gain of the controller that should be designed to minimize the following cost function:

$$J(u) = \int_0^{\infty} \{z^T Q z + u^T R u\} dt, \quad (8)$$

where $z = [\tilde{x} \quad x_i]^T$, R and Q are positive-semidefinite and positive-definite matrices which are determined by an expert. In fact, these matrices determine a trade-off between cost of control effort and tracking performance. Therefore, the performance of the controller is sensitive to these matrices. It should be noted that there is a straightforward solution to determine K based on solving a differential equation named Riccati Differential Equation (RDE).

To check the performance of the proposed controller, a simulation has been performed. In this simulation, a set of reference trajectories has been generated and the controller should track this reference in presence of measurement noise. The simulation results are shown in Figure 5. The results showed that the controller is able to track the references even when the measurements are affected by noise. In the next section, we explain how the reference trajectories are generated.

5. Reference Trajectories Planner

Our walking reference trajectories planner is composed of five sub planners which are connected together hierarchically. The first level of this hierarchy is a footstep planner which generates a set of foot positions based on given step information, terrain information and some predefined constraints (e.g., maximum and minimum step length, step width, distance between feet, etc.). To do so, we consider a state variable to represent the current state of the robot's feet:

$$s = (x_l, y_l, \theta_l, \phi_l, x_r, y_r, \theta_r, \phi_r) \quad (9)$$

where $x_l, y_l, \theta_l, x_r, y_r, \theta_r$ are the position and orientation of the left and right feet, respectively. ϕ_l, ϕ_r represent the current state of feet which is 1 if the foot is the swing foot and -1 otherwise. Walking is a period motion which is generated by moving the right and the left legs alternating. Therefore, we parametrize a step action by a length and an angle from the swing foot position at the beginning of steps $a = (R, \sigma)$. According to the input parameters and the current state of the feet, an action should be taken and the state transits to a new state, $s' = t(s, a)$. Afterwards, the current footstep will be saved ($f_i \quad i \in \mathbb{N}$) and ϕ_l and ϕ_r will be toggled.

The second planner is the ZMP planner that uses the planned footstep information to generate ZMP reference trajectories. In our target framework, the ZMP reference planner is formulated as follows:

$$r_{zmp} = \begin{cases} \begin{cases} f_{i,x} \\ f_{i,y} \end{cases} & 0 \leq t < T_{ss} \\ \begin{cases} f_{i,x} + \frac{L_{sx} \times (t - T_{ss})}{T_{ds}} \\ f_{i,y} + \frac{L_{sy} \times (t - T_{ss})}{T_{ds}} \end{cases} & T_{ss} \leq t < T_{ss} + T_{ds} \end{cases}, \quad (10)$$

where $f_i = [f_{i,x} \quad f_{i,y}]$ are the planned footsteps on a 2D surface ($i \in \mathbb{N}$), L_{sx} and L_{sy} represent the step length and width, T_{ss} , T_{ds} are the single support and double support durations, respectively, and t is the time which will be reset at the end of each step ($t \geq T_{ss} + T_{ds}$). The third planner is the swing leg planner which generates the swing leg trajectory using a cubic spline function. This planner uses three control points that

are the position of the swing leg at the beginning of the step, the next footstep position and a point between them with a predefined height (Z_{swing}). The fourth planner is the global sinusoidal planner which generates three sinusoidal trajectories for the COM height, the torso angles and the arm positions. The fifth planner is the hip planner which uses the generated ZMP and torso trajectories to generate hip trajectory. Indeed, these trajectories are used to determine the positions of the hip at the beginning and the end of step. Using these positions, Equation (4) can be solved as a boundary value problem as follows:

$$x(t) = g_x + \frac{(g_x - x_f) \sinh(\sqrt{\mu}(t - t_0)) + (x_0 - g_x) \sinh(\sqrt{\mu}(t - t_f))}{\sinh(\sqrt{\mu}(t_0 - t_f))}, \quad (11)$$

where $g_x = r_{zmp_x} - \frac{\alpha l}{1 + \alpha} \theta_{to} + \frac{\alpha \beta l}{\mu(1 + \alpha \beta)} \ddot{\theta}_{to}$, t_0 , t_f , x_0 , x_f are the times and corresponding positions of the hip at the beginning and at the end of a step, respectively. In this work, T_{ds} is considered to be zero, which means ZMP transits to the next step at the end of each step instantaneously. Moreover, x_f is assumed to be in the middle of current support foot and next support foot ($\frac{f_i + f_{i+1}}{2}$).

6. Learning Framework

Our learning framework employs the Proximal Policy Optimization (PPO) algorithm, introduced by Schulman et al. [37], which was chosen due to its success in optimizing low-level skills concerning the NAO robot [21, 38, 39, 40, 41], and high-level skills [42], where it outperformed other algorithms such as TRPO or DDPG. The chosen implementation uses the clipped surrogate objective:

$$L(\theta) = \hat{\mathbb{E}}_t [\min(r_t(\theta) \hat{A}_t, \text{clip}(r_t(\theta), 1 - \epsilon, 1 + \epsilon) \hat{A}_t)],$$

$$\text{where } r_t(\theta) = \frac{\pi_\theta(a_t | s_t)}{\pi_{\theta_{old}}(a_t | s_t)}, \quad (12)$$

where \hat{A}_t is an estimator of the advantage function at timestep t . The clip function clips the probability ratio $r_t(\theta)$ in the interval given by $[1 - \epsilon, 1 + \epsilon]$. This implementation alternates between sampling data from multiple parallel sources, and performing several epochs of stochastic gradient ascent on the sampled data, to optimize the objective function.

The clipping parameter ε was set to 0.2, as suggested by Schulman et al. [37]. Also, as in the implementations published by OpenAI for the 3D humanoid environment [33], the entropy bonus was not used, and the number of optimization epochs and batches was set to 10 and 64, respectively. Some other hyperparameters were tuned using grid search: step size (2.5×10^{-4}); batch size (4096); and the Adaptive Moment Estimation (Adam) [43] optimizer was set to use a constant scheduler. Finally, the Generalized Advantage Estimation (GAE) [44] algorithm’s parameters — gamma and lambda — were set to 0.99 and 0.95, respectively, accordingly to the ranges established by the GAE’s authors as best-performing for 3D biped locomotion.

The policy is represented by a multilayer perceptron with two hidden layers of 64 neurons. The number of inputs, outputs, and the maximum number of time steps for the optimization are dependent on the scenario and will be described in section 8.3. The training session was parallelized to improve the optimization duration.

7. Overall Structure

In this section, the previously introduced planner and controller will be coupled together to generate stable locomotion. To do that, we designed a modular framework composed of six main modules. The overall architecture of this framework is depicted in Figure 6. As shown in this figure, the walking process is controlled by a state machine which abstracts the process into four distinct states: *Idle*, *Initialize*, *Single Support* and *Double Support*. In this state machine, the transitions are triggered by a timer that is associated to each state. Additionally, it can be triggered by an emergency signal generated according to the controller’s state in key moments, such as when a swift move is necessary to regain equilibrium after a strong external perturbation. The *Idle state* is the initial state in which the robot is standing in place and waiting to receive a walking signal, which can be generated by an operator or a path planning algorithm. That signal triggers the *Initialize state*, in which the walking parameters and configurations are loaded from a data base. Afterwards, the robot is ready to walk by shifting its COM towards the first support foot. The next state is triggered after a predefined time. During *Single Support State* and *Double Support State*, the dynamics planner

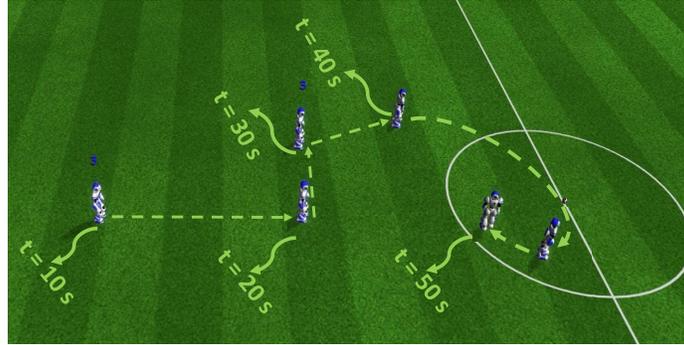


Figure 7: Omnidirectional walk scenario. In this scenario, the simulated robot should follow the commands that are generated by an operator: Green-dashed arrows show a set of commands that has been generated for this simulation, including forward walk, side walk, diagonal walk and turning while performing diagonal walking.

the official RoboCup 3D simulation environment which is based on SimSpark, a multi-agent simulator. This simulator relies on the Open Dynamics Engine (ODE) to simulate rigid body dynamics. The physics engine is updated every 0.02s. The simulator can also be configured to update the physics engine just after receiving commands from all agents. This greatly improves simulation speed and provides a better environment for learning approaches.

8.1. Omnidirectional Walk

This scenario is designed to demonstrate the performance of the framework in providing an omnidirectional walk. The simulated robot starts from an idle state and follows a command comprising length (X), width (Y) and angle (α) of the step, which is determined by an operator. Note that the step time is constant and set to 0.2s. To avoid discontinuity in the input command, a first-order lag filter is used, yielding a smooth transition.

At the beginning of this scenario, the robot is walking in place and all the setpoints are zero ($X = 0.0\text{m}$, $Y = 0.0\text{m}$, $\alpha = 0.0\text{deg/s}$). At $t = 10\text{s}$, the operator sets the step length ($X = 0.05\text{m}$) to generate forward walking; at $t = 20\text{s}$, the operator resets the step length ($X = 0.0\text{m}$) and sets the step width ($Y = 0.04\text{m}$) to generate side walking; at $t = 30\text{s}$, while the robot is performing side walking, the operator sets the step

length ($X = 0.05\text{m}$) to generate diagonal walking; at $t = 40\text{s}$, while the robot is performing diagonal walking, the robot is commanded to turn right simultaneously, by setting the step angle ($\alpha = 10\text{deg}$); and finally, at $t = 60\text{s}$, all the set points are reset and the robot starts walking in place. A set of snapshots of this simulation is depicted in Figure 7. The simulation results showed that the framework was able to generate omnidirectional walking according to the input commands. A video of this simulation is available online at: <https://www.dropbox.com/s/32gml9mtumps1np/OmniWalkRC.mp4?dl=0>.

8.2. Optimizing the Walking Parameters

This scenario is focused on optimizing the walking parameters to generate the fastest stable forward walk. In this scenario, the robot is placed 10m away from the halfway line and it should walk straight forward towards that line as fast as possible. Initially, the best parameters were hand-tuned and, after several attempts, the maximum walking speed did not exceed 53cm/s. Afterwards, based on the parametric nature of the proposed planner and controller, a GA is used to optimize the parameters to improve the walking speed. To do that, 8 parameters of the framework have been selected to be optimized. These parameters are the step length (x), step width (y), step angle (α), height of the swing leg (z_{sw}), duration of a step (T_{ss}), torso inclination TI_{to} , amplitude of the COM (A_z) and amplitude of the torso movement (A_{to}). In our optimization scenario, the simulated robot should walk forward for 10 seconds and its performance will be evaluated based on the following cost function:

$$f(\phi) = -|\delta x| + |\delta y| + \epsilon \quad (13)$$

where ϕ represents the selected parameters, δX , δY are the distance covered in X-axis and Y-axis, respectively, ϵ is used to penalize the robot when it falls during walking ($\epsilon = 100$) otherwise it is zero. According to this cost function, the simulated robot is rewarded for straight forward walk and it is penalized for deviation and falling. A slow and stable forward walking (0.11m/s) is used as an initial solution to start the optimization process. It should be noted that, each iteration has been repeated three

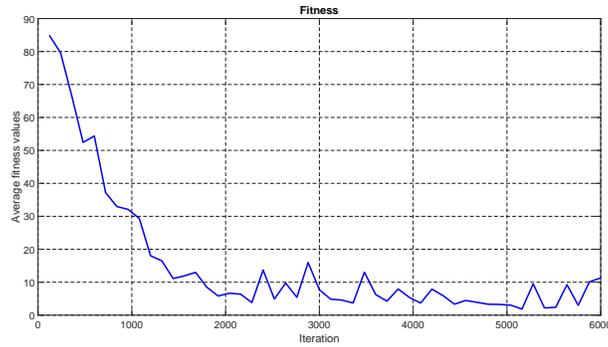


Figure 8: Evolution of the fitness.

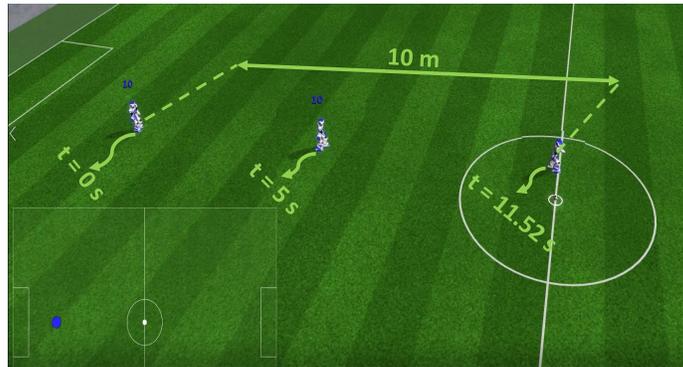


Figure 9: The optimization scenario and the results of an exemplary test after optimizing the parameters. In this test, the simulated robot has been placed at a specific point which is 10m far from the center of the field and it should walk towards the center as fast as possible. The results showed that the robot touched the midline at $t = 11.52s$.

times and the average of the fitnesses was used to be sure about the walking performance. The fitness values have been recorded for each iteration and the average fitness values can be visualized in Figure 8. The average fitness value starts at around 85 and after about 2000 iterations, it drops under 10, which is much better than the first solution. The optimization has been executed for 6000 iterations. After optimizing the parameters, the walking velocity reaches 0.866m/s, which is 61% faster than the best hand-tuned solution. The best parameters found by the GA are shown in Table 1. The optimization scenario and a set of snapshots of a test are shown in Figure 9. A video of this simulated scenario is available online at: <https://www.dropbox.com/s/>

Table 1: The best parameters.

Parameter	Symbol	Value
Step duration	T_{ss}	0.1274s
Step length	x	0.09059m
Step width	y	0.010086m
Step angle	α	-0.2899 deg
Height of swing	z_{sw}	0.038m
Torso inclination	TI_{to}	5.601 deg
Amplitude of height of COM	A_z	-0.004
Amplitude of torso movement	A_{to}	-1.9195

wm5y8dkekd2fnpo/OmniWalkRCOptimized.mp4?dl=0.

To compare the effectiveness of the dynamics model, this scenario has been repeated for the dynamics models (a) and (b) presented in Figure 2. To do so, the planner and the controller have been adjusted according to the dynamics models and then their parameters have been refined manually. Finally, this simulation scenario has been repeated to find the maximum forward speed of each model. The simulation results are summarized in Table 2. The simulation results validated that the sinusoidal motion of the height of COM improves the stability and allows the robot to move faster in comparison with fixed COM.

Table 2: Summary of the results in the maximum speed scenario.

Dynamics model	maximum speed
LIPM	0.590 m/s
LIPM + vertical motion of COM	0.630 m/s
LIPM + vertical motion of COM + Torso	0.866 m/s

8.3. Learning to Improve the Upper Body Efficiency

This scenario was designed to improve the efficiency of the walking gait in terms of speed and stability during the most common walking patterns – forward walking and turning. To mitigate the effects of the learning process on the maneuverability and



Figure 10: Stability optimization scenario. The robot exploits the most common walking patterns – forward walking and turning – to keep itself within a predefined squared area of side length 2m. When inside that area, the robot walks forward as fast as possible. Otherwise, it turns at a random rate until it is directed towards the area.

predictability of the walking trajectory, only the arms actuators and COM height were optimized. The robot is initially placed in an arbitrary position within a squared area of side length 2m, as depicted in Figure 10. It then starts walking forward with the best parameters found in Section 8.2. When the robot steps out of the predefined area, it starts to turn in either direction at a random rate $|\alpha| \in [30 \text{ deg/s}, 60 \text{ deg/s}]$, until it is facing the square again. This process is repeated continuously until the episode ends with the robot falling (detected when its z coordinate drops below 0.3m). The fact that the robot runs at full speed when changing direction, and that it needs to constantly adapt to different turning rates makes this a very challenging scenario.

This optimization problem can be formalized as a Markov Decision Process (MDP) – a 4-tuple $\langle S, A, p, R \rangle$ – where S denotes the set of states, A the set of possible actions, and R the set of possible numerical rewards. The dynamics of the MDP is given by the state-transition probability function $p(s'|s, a) : S \times S \times A(s) \rightarrow [0, 1]$ which gives the probability of ending in state s' given the current state s and action a .

The interaction between agent and environment is performed at discrete time steps ($t = 0, 0.02, 0.04, \dots$). The robot's behavior was optimized by the PPO algorithm, using 67 observed variables, as listed in Table 3. The first parameter indicates the current

Table 3: State space for the stability optimization.

Parameter	Data size ($\times 32\text{b}$)
α	1
Gyroscope	3
Accelerometer	3
Joints Position	20
Joints Speed	20
Controller Actions	20

turning rate. The inertial sensors (gyroscope and accelerometer) are both composed of 1 variable per axis in a three-dimensional space. The position and speed of all joints (excluding the head) is important to obtain a correct state representation, even though the action space only controls a limited number of these joints. Finally, the joint positions computed by the analytical controller are fed to the algorithm, and later added as residuals to the output values. These positions can be used by the network to predict the next analytical state, so that the produced residuals can be adjusted accordingly. In preliminary tests, removing this information from the state space results in a loss of performance between 5% and 20%, depending on possible action space combinations.

The action space encompasses four angle variables per arm (shoulder roll, shoulder pitch, elbow yaw, elbow roll) and one variable to define the setpoint of the COM height at each step. The arm joints angles are computed by summing the analytical controller’s output to the neural network’s corresponding output. This forms a controller which uses the planner’s arms control signals both as state data and action bias.

The objective of this scenario is to improve forward speed and stability at all times (i.e., when moving forward or turning). The former requirement is met by rewarding the agent for stepping forward and not sideways, which can be numerically translated into the scalar projection of its velocity vector \vec{v} in the direction of its orientation unit vector \vec{o} . Let $\vec{v} = P_t - P_{t-1}$, where P_t and P_{t-1} are the current and previous positions of the robot, respectively. The partial reward to motivate forward speed is then $\max(\vec{v} \cdot \vec{o}, 0)$, where \cdot denotes the dot product. The minimum reward value is limited to zero because walking backward or sideways is not worse than falling. The second

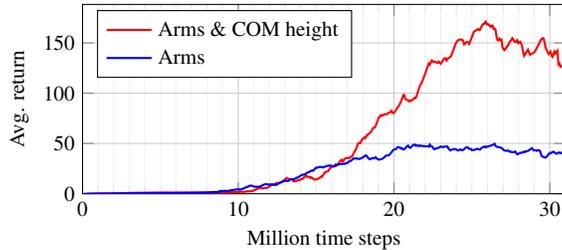


Figure 11: Learning curves considering only the arms (blue) and considering arms and COM height (red).

requirement — stability — is motivated by a constant k , set empirically to 0.01, that rewards the agent for staying alive. More precisely, it favors stability at the cost of lowering the speed. The complete immediate reward can then be formulated as:

$$r = \max(\vec{v} \cdot \vec{\sigma}, 0) + k. \quad (14)$$

The learning algorithm was first applied to the arms actuators and later extended to the COM height. Figure 11 shows the average return evolution when learning only the arms (blue line) and after adding the COM height (red line). The former optimization plateaued at around 20M time steps, and the latter at around 26M time steps. It is important to note that the return obtained during the optimization was based on a stochastic policy whereas in the following tests, we used the corresponding deterministic policy.

The results were divided into two sections: Original scenario – the robot is tested in the same scenario used for learning (see Figure 10) and the analysis delves into the same metrics used to define the reward function; Straightforward path – the robot’s direction is constantly corrected to describe a linear path and the resulting behavior is analyzed in terms of efficiency.

8.3.1. Original Scenario Results

The robot was evaluated with regard to stability and speed in the same scenario where the learning algorithm was applied. Stability was measured by the episode length, since it terminates once the robot falls to the ground. No time limitation was imposed per episode. Speed was measured at every iteration and averaged at the end of

Table 4: Original scenario results – average speed and duration

Parameter	Ep. Length Mean \pm SD (s)	Speed Mean \pm SD (m/s)
Baseline	5.1 \pm 2.9	0.602 \pm 0.027
Arms	51.6 \pm 31.8	0.710 \pm 0.037
Arms & COM height	148.2 \pm 153.0	0.956 \pm 0.060

the episode. Table 4 lists the average speed and duration results for 500 episodes. The first line corresponds to the walk optimized in Section 8.2, which is used as a baseline. The robot walks on average for 5.1 seconds with a standard deviation (SD) of 2.9s before falling, generally on the first or second sharp change of direction. The mean speed, from a stand-still position to the end of each episode, was 0.602m/s with a SD of 0.027m/s.

After learning how to control the arms, the episode duration increased tenfold, on average, and the mean speed rose to 0.710m/s. Most falls occur at an advanced stage or during the initial sharp turns, hence the larger standard deviation. Adding the COM height to the group of controlled variables increased the episode length to almost 15 times the initial value. When compared to the version without the COM height adaptation, this metric improved approximately 3 times. The mean speed rose to 0.956m/s, a gain of almost 60% in relation to the baseline. In every episode the robot eventually falls. As aforementioned, when the robot steps out of the predefined area, it starts to turn with a random turning rate between 30 and 60deg/s, in either direction. Most falls occur when approximating the upper limit or when the rotation direction is inverted after the robot enters and exits the predefined area in a short time (e.g. when stepping over a corner of that area). A video comparing the baseline with the Arms & COM height optimization is available online at: https://www.dropbox.com/s/d1o74tpb2u6tr9f/OmniWalkLearningComparison_subtitle.mp4?dl=0.

8.3.2. *Straightforward Path Results*

To evaluate the gait efficiency without taking stability into account, the displacement of certain joints as well as the average speed were analyzed, as discussed later

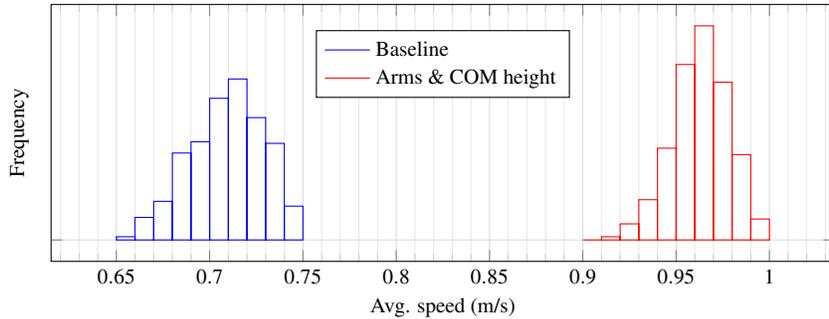


Figure 12: Mean speed comparison between the baseline (on the left) with the best optimization using the Arms & COM Height controller (on the right). These values were averaged for 500 successful episodes, where the robot runs for 12m in a straight line.

in this section. Due to the challenging nature of the learning scenario, and to provide a fair comparison with the baseline algorithm, a simplified setup was developed. The objective is to compare the baseline and best optimization algorithms while the robot tries to describe a straight path of 12m length. The turning parameter is computed at every iteration to maximize the path’s linearity using a reactive proportional controller. After 12m, if the robot has not fallen, the episode is considered successful.

Figure 12 compares the baseline’s mean speed for the entire path with its improved version using the Arms & COM height controller. In both cases, the speed values were averaged for 500 successful episodes. In comparison with Table 4, the baseline algorithm improved its mean speed from 0.602m/s to 0.704m/s. The Arms & COM height optimization went from 0.956m/s to 0.958m/s, indicating that the robot has a virtually constant speed, whether turning or not. The improvement of the optimized version in relation to the baseline is about 36%.

The total angular displacement performed by certain groups of joints during a successful episode was analyzed, as this metric provides a reasonable indicator of energy consumption, considering that the actuators load is not disclosed by the server. Figure 13 compares the displacement sum of the arms joints with the displacement sum of all robot joints (except for the head). This analysis was performed for different stages of the linear path described by the robot. In the first meter, as expected, the robot spends more energy while gaining momentum, and then it stabilizes. Considering only

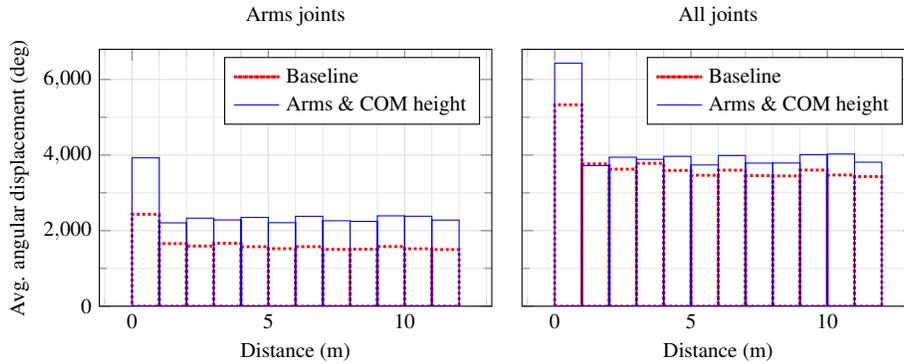


Figure 13: Angular displacement performed by all arms joints (sum) during an episode, averaged for 500 successful episodes (on the left). The linear path described by the robot was divided into sections of 1m, which are represented by each bar. The baseline is represented by the dotted red bars while the optimized version is represented by the solid blue bars. The same sort of analysis for all joints is depicted on the right.

the arms joints (shoulder roll, shoulder pitch, elbow yaw, elbow roll), the average angular displacement for the entire episode rose 49%. Despite this result, the same analysis performed for all joints yields an increment of only 10%. Therefore, without considering stability gains, the ratio of relative speed improvement to relative displacement increment in successful episodes is 3.6. In essence, the robot became much more energy efficient, as a small raise in energy consumption led to a considerably faster gait.

9. Discussion

Simulation results showed that the framework is able to generate a fast and stable omnidirectional walk and improve its performance by learning how to control the arms and the height of the COM. Indeed, the results showed that providing a tight coupling between analytical approaches and ML improves the performance considerably. In the remaining of this section, we point out the features and limitations of the proposed framework and provide comparisons with the results of previous works.

9.1. Features

- **Architecture:** the modular architecture of the proposed framework provides some important properties such as reducing the complexity and increasing the

flexibility. In comparison with approaches that are based on heuristic methods [45, 46, 12, 13, 14, 15] or based on learning from scratch [21, 12, 20], our framework is able to migrate to different humanoid platforms with small changes to the control module.

- **Computational efficiency:** unlike the approaches presented in [6, 27, 47, 48, 49] which are based on online optimization (e.g., MPC), our controller was designed on top of an offline optimization algorithm. Therefore, it does not need powerful computational resources and it can be deployed on any platform easily.
- **Considering the upper body dynamics:** most of the presented approaches in the literature used LIPM as their dynamics model, mainly due to its linearity and simplicity. Unlike LIPM-based approaches, we take into account the robot's upper body dynamics and we showed how this consideration helps to enhance the stability and speed of the robot, while improving the energy efficiency as a ratio of mean speed to total angular displacement.
- **Release the height of COM constraint:** LIPM-based approaches assume a fixed vertical position for the COM. According to this assumption, the knee joints have to be bent while the robot is walking, which is harmful for the knee joints and causes additional energy consumption. Additionally, walking with bent knees is not very human-like. We released this constraint by assuming a sinusoidal movement for the vertical position of the COM. We showed that this assumption not only cancels the explained limitations but it also improves the stability.
- **Performance:** to have an entirely fair comparison, the performance of our framework should be compared with other frameworks in the same scenario and simulator. To do so, we took into consideration the maximum forward speed, and our proposed framework provides a faster walk than the agents in [45, 50, 46, 25, 19] and slower than [51, 21, 38, 41]. However, some of the faster examples are solely focused on sprinting forward, without the basic ability of changing direction [21, 38, 41]. The comparison results are summarized in Table 5.

Table 5: Comparison Results

	Maximum speed	Ability of changing direction
[51]	1.180 m/s	YES
Proposed framework	0.956 m/s	YES
[19]	0.805 m/s	YES
[46]	0.770 m/s	YES
[25]	0.590 m/s	YES
[38]	3.910 m/s	NO
[21]	2.500 m/s	NO
[41]	1.340 m/s	NO
[50]	0.550 m/s	NO
[45]	0.510 m/s	NO

- Learning flexibility:** we believe that a humanoid robot should be able to learn from experience, not only to create a new behavior but also to improve its skills. Additionally, it should be able to reuse its knowledge in different scenarios. Learning how to control the arms and the COM height had a positive effect under different conditions in which the robot was not explicitly trained. The robot preserved its stability and speed when subjected to constant orientation adjustments to move in a straight line. Furthermore, we kept the learning module on top of the others to allow situations where generalization is not a conceivable solution. This is an improvement over learning from scratch approaches, as it builds upon a logical and reliable initial solution. This analytical layer is less prone to modeling errors than the learning layer, which is critical when transferring the knowledge to a real robot. After tuning the control module to new conditions, the neural network can be partially retrained by leveraging existing knowledge of similar tasks. This architecture allows for a plethora of modular optimizations aimed at stability, speed, energy efficiency, path optimization, context awareness problems (including prevention and recovery), etc.
- Controller and Robustness:** some approaches [22, 25] used a dynamics model just to generate a feed-forward walk and did not consider any controller to track

Table 6: Summary of the results in the maximum speed scenario.

Dynamics model	ML algorithm	maximum speed
LIPM	GA	0.590 m/s
LIPM + vertical motion of COM	GA	0.630 m/s
LIPM + vertical motion of COM+ Torso	GA	0.866 m/s
LIPM	GA+PPO	0.710 m/s
LIPM + vertical motion of COM	GA+PPO	0.741 m/s
LIPM + vertical motion of COM+ Torso	GA+PPO	0.956 m/s

the references. Other approaches that are based on learning from scratch [12, 20, 21] do not take into account any controller explicitly. Instead, they use a learning algorithm to develop a controller implicitly. Unlike these approaches, we believe that a robust controller is an essential module of a walking framework due to the unstable nature of a humanoid robot. More specifically, when deploying the framework on a real robot, using a closed-loop walking is the best approach because it provides a better stability guarantee. Moreover, as we showed, the ML algorithms can be used on top of this controller to improve its performance. The summary of the results in the maximum speed scenario are presented in the Table 6.

9.2. Limitations

- **Swing leg dynamics:** the legs of a humanoid robot are generally composed of six joints and have non-negligible masses. In our dynamics models, the swing leg is considered to be massless, which affects the controller performance. Taking into account the inertia and mass of the swing leg can minimize tracking errors and improve the controller’s performance [5, 52].
- **Reality Gap:** the disparity between reality and simulation is a matter of concern when employing offline ML techniques. Learning to improve the upper body efficiency took between 20 and 26 million time steps. Other works have shown the optimization of robotic tasks, such as squatting [27], using RL combined with an analytical controller, in under 10M time steps. Haarnoja et al. also demon-

strated that learning humanoid tasks from scratch can also be performed in about the same period of time [53]. However, it must be noted that these approaches employed distinct environments with different robots, directly influencing the complexity of the task. Learning to run using the NAO robot in SimSpark can take close to 200M time steps [38, 21]. Nevertheless, 20-26 million time steps can still be characterized as poor sample efficiency, as it takes a considerable amount of time and must be performed in a simulated environment. The gap between both worlds largely affects the transferability of knowledge to the real robot. Despite the scientific community’s considerable effort to reduce this gap, it remains an issue when dealing with intricate robot models. Additionally, it is not possible to learn directly on the real robot due to the high potential of mechanical damage.

10. Conclusion

In this paper, we have tackled the problem of developing a robust biped locomotion framework by proposing a tight coupling between an analytical control approach and a reinforcement learning approach. The overall architecture of the framework was composed of six distinct modules which were hierarchically structured. We abstracted the overall dynamics of a humanoid robot into two masses. Then, we used the ZMP concept and some assumptions to represent this dynamics model as a linear state space system. The system was composed of four states and we explained how it can be used to plan and control the walking reference trajectories. Particularly, the planner was composed of five sub planners and the controller was formulated as an LQG controller, which is not only robust against uncertainties but also provides a promising solution using an offline optimization. We analyzed the performance of the controller in the presence of uncertainties using simulations and the results validated its performance. Moreover, we illustrated how the parametric nature of the framework allows us to use the PPO algorithm on top of an analytical control approach to improve the performance of the framework. Finally, the performance of the proposed framework was validated in several simulated scenarios. The first two scenarios were focused on examining

the ability of the framework in generating an omnidirectional walk and finding the maximum velocity of the forward walk. The third scenario was designed to assess the capability of the learning module in improving the framework's performance. The robot learned how to move its arms and COM height in order to improve the stability, speed and energy efficiency. This limited action space enabled the robot to learn how to walk without falling for much longer periods (almost 15 times longer), while also improving the speed by 60% when walking forward or turning.

As future work, we would like to design a more accurate dynamics model by considering the mass of the swing leg to improve the framework's performance. Additionally, we would like to extend our framework by adding another module to handle the emergency conditions based on learning a set of specific actions.

Acknowledgement

This research is supported by Portuguese National Funds through Foundation for Science and Technology (FCT) through FCT scholarship SFRH/BD/118438/2016. The second author is supported by FCT under grant SFRH/BD/139926/2018.

References

- [1] S. Kajita, K. Tani, Study of dynamic biped locomotion on rugged terrain-derivation and application of the linear inverted pendulum mode, in: *Robotics and Automation, 1991. Proceedings., 1991 IEEE International Conference on, IEEE, 1991*, pp. 1405–1411.
- [2] S. Kajita, F. Kanehiro, K. Kaneko, K. Fujiwara, K. Harada, K. Yokoi, H. Hirukawa, Biped walking pattern generation by using preview control of zero-moment point, in: *Robotics and Automation, 2003. Proceedings. ICRA'03. IEEE International Conference on, Vol. 2, IEEE, 2003*, pp. 1620–1626.
- [3] S. Kajita, M. Morisawa, K. Miura, S. Nakaoka, K. Harada, K. Kaneko, F. Kanehiro, K. Yokoi, Biped walking stabilization based on linear inverted pendulum

- tracking, in: *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*, IEEE, 2010, pp. 4489–4496.
- [4] S. Shimmyo, T. Sato, K. Ohnishi, Biped walking pattern generation by using preview control based on three-mass model, *Industrial Electronics, IEEE Transactions on* 60 (11) (2013) 5137–5147.
- [5] S. Faraji, A. J. Ijspeert, 3LP: A linear 3D-walking model including torso and swing dynamics, *the international journal of robotics research* 36 (4) (2017) 436–455.
- [6] R. J. Griffin, G. Wiedebach, S. Bertrand, A. Leonessa, J. Pratt, Walking stabilization using step timing and location adjustment on the humanoid robot, atlas, in: *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), IEEE, 2017*, pp. 667–673.
- [7] M. Kasaei, N. Lau, A. Pereira, A robust biped locomotion based on linear-quadratic-gaussian controller and divergent component of motion, in: *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems, IEEE, 2019*, pp. 1429–1434.
- [8] M. M. Kasaei, N. Lau, A. Pereira, A model-based biped walking controller based on divergent component of motion, in: *2019 IEEE International Conference on Autonomous Robot Systems and Competitions (ICARSC), IEEE, 2019*, pp. 1–6.
- [9] J. Yamaguchi, E. Soga, S. Inoue, A. Takanishi, Development of a bipedal humanoid robot-control method of whole body cooperative dynamic biped walking, in: *Proceedings 1999 IEEE International Conference on Robotics and Automation (Cat. No. 99CH36288C), Vol. 1, IEEE, 1999*, pp. 368–374.
- [10] O. Khatib, L. Sentis, J.-H. Park, A unified framework for whole-body humanoid robot control with multiple constraints and contacts, in: *European Robotics Symposium 2008, Springer, 2008*, pp. 303–312.

- [11] K. Ishihara, T. D. Itoh, J. Morimoto, Full-body optimal control toward versatile and agile behaviors in a humanoid robot, *IEEE Robotics and Automation Letters* 5 (1) (2019) 119–126.
- [12] J. Shan, C. Junshi, C. Jiapin, Design of central pattern generator for humanoid robot walking based on multi-objective ga, in: *Proceedings. 2000 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2000)*(Cat. No. 00CH37113), Vol. 3, IEEE, 2000, pp. 1930–1935.
- [13] J. Lee, K. Seo, Generation of walking trajectory of humanoid robot using cpg, *Journal of Korean Institute of Intelligent Systems* 23 (4) (2013) 360–365.
- [14] C. Liu, D. Wang, Q. Chen, Central pattern generator inspired control for adaptive walking of biped robots, *IEEE Transactions on Systems, Man, and Cybernetics: Systems* 43 (5) (2013) 1206–1215.
- [15] J. Yu, M. Tan, J. Chen, J. Zhang, A survey on cpg-inspired control models and system implementation, *IEEE transactions on neural networks and learning systems* 25 (3) (2013) 441–456.
- [16] P. A. Guertin, The mammalian central pattern generator for locomotion, *Brain research reviews* 62 (1) (2009) 45–56.
- [17] G. Zhong, N. A. Shevtsova, I. A. Rybak, R. M. Harris-Warrick, Neuronal activity in the isolated mouse spinal cord during spontaneous deletions in fictive locomotion: insights into locomotor central pattern generator organization, *The Journal of physiology* 590 (19) (2012) 4735–4759.
- [18] E. Menelaou, D. L. McLean, Hierarchical control of locomotion by distinct types of spinal v2a interneurons in zebrafish, *Nature communications* 10 (1) (2019) 1–12.
- [19] M. Kasaei, N. Lau, A. Pereira, A fast and stable omnidirectional walking engine for the nao humanoid robot, in: S. Chalup, T. Niemueller, J. Suthakorn, M.-A. Williams (Eds.), *RoboCup 2019: Robot World Cup XXIII*, Springer International Publishing, Cham, 2019, pp. 99–111.

- [20] G. Endo, J. Morimoto, T. Matsubara, J. Nakanishi, G. Cheng, Learning cpg-based biped locomotion with a policy gradient method: Application to a humanoid robot, *The International Journal of Robotics Research* 27 (2) (2008) 213–228.
- [21] M. Abreu, L. P. Reis, N. Lau, Learning to run faster in a humanoid robot soccer environment through reinforcement learning, in: *Robot World Cup*, Springer, 2019, pp. 3–15.
- [22] P. MacAlpine, S. Barrett, D. Urieli, V. Vu, P. Stone, Design and optimization of an omnidirectional humanoid walk: A winning approach at the robocup 2011 3d simulation competition, in: *Twenty-Sixth AAAI Conference on Artificial Intelligence*, 2012.
- [23] J. Or, A hybrid cpg–zmp control system for stable walking of a simulated flexible spine humanoid robot, *Neural Networks* 23 (3) (2010) 452–460.
- [24] B. He, Z. Wang, R. Shen, S. Hu, Real-time walking pattern generation for a biped robot with hybrid cpg-zmp algorithm, *International Journal of Advanced Robotic Systems* 11 (10) (2014) 160.
- [25] S. M. Kasaei, D. Simões, N. Lau, A. Pereira, A hybrid zmp-cpg based walk engine for biped robots, in: *Iberian Robotics conference*, Springer, 2017, pp. 743–755.
- [26] J. Carpentier, S. Tonneau, M. Naveau, O. Stasse, N. Mansard, A versatile and efficient pattern generator for generalized legged locomotion, in: *2016 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2016, pp. 3555–3561.
- [27] I. Koryakovskiy, M. Kudruss, H. Vallery, R. Babuška, W. Caarls, Model-plant mismatch compensation using reinforcement learning, *IEEE Robotics and Automation Letters* 3 (3) (2018) 2471–2477.
- [28] K.-T. Song, C.-H. Hsieh, Cpg-based control design for bipedal walking on unknown slope surfaces, in: *2014 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2014, pp. 5109–5114.

- [29] M. Missura, S. Behnke, Gradient-driven online learning of bipedal push recovery, in: 2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), IEEE, 2015, pp. 387–392.
- [30] A. Massah, A. Zamani, Y. Salehinia, M. A. Sh, M. Teshnehlab, A hybrid controller based on cpg and zmp for biped locomotion, *Journal of Mechanical science and technology* 27 (11) (2013) 3473–3486.
- [31] J. Liu, O. Urbann, Bipedal walking with dynamic balance that involves three-dimensional upper body motion, *Robotics and Autonomous Systems* 77 (2016) 39–54.
- [32] A. Abdolmaleki, D. Simoes, N. Lau, L. P. Reis, G. Neumann, Contextual relative entropy policy search with covariance matrix adaptation, in: 2016 International Conference on Autonomous Robot Systems and Competitions (ICARSC), IEEE, 2016, pp. 94–99.
- [33] P. Dhariwal, C. Hesse, O. Klimov, A. Nichol, M. Plappert, A. Radford, J. Schulman, S. Sidor, Y. Wu, P. Zhokhov, Openai baselines, <https://github.com/openai/baselines> (2017).
- [34] M. Vukobratovic, A. Frank, D. Juricic, On the stability of biped locomotion, *IEEE Transactions on Biomedical Engineering BME-17* (1) (1970) 25–36.
- [35] D. A. Winter, G. K. Ruder, C. D. MacKinnon, Control of balance of upper body during gait, in: *Multiple muscle systems*, Springer, 1990, pp. 534–541.
- [36] S. Kajita, M. Benallegue, R. Cisneros, T. Sakaguchi, M. Morisawa, H. Kaminaga, I. Kumagai, K. Kaneko, F. Kanehiro, Position-based lateral balance control for knee-stretched biped robot, in: 2019 IEEE-RAS 19th International Conference on Humanoid Robots (Humanoids), IEEE, 2019, pp. 17–24.
- [37] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, O. Klimov, Proximal policy optimization algorithms, *CoRR* 1707.06347.

- [38] L. Carvalho Melo, M. R. Omena Albuquerque Máximo, Learning humanoid robot running skills through proximal policy optimization, in: 2019 Latin American Robotics Symposium (LARS), 2019 Brazilian Symposium on Robotics (SBR) and 2019 Workshop on Robotics in Education (WRE), 2019, pp. 37–42.
- [39] H. Teixeira, T. Silva, M. Abreu, L. P. Reis, Humanoid robot kick in motion ability for playing robotic soccer, in: 2020 IEEE International Conference on Autonomous Robot Systems and Competitions (ICARSC), IEEE, 2020, pp. 34–39.
- [40] D. C. Melo, M. R. O. A. Máximo, A. M. da Cunha, Push recovery strategies through deep reinforcement learning, in: 2020 Latin American Robotics Symposium (LARS), 2020 Brazilian Symposium on Robotics (SBR) and 2020 Workshop on Robotics in Education (WRE), IEEE, 2020, pp. 240–245.
- [41] M. Abreu, N. Lau, A. Sousa, L. P. Reis, Learning low level skills from scratch for humanoid robot soccer using deep reinforcement learning, in: 2019 IEEE International Conference on Autonomous Robot Systems and Competitions (ICARSC), IEEE, 2019, pp. 1–8.
- [42] A. F. V. Muzio, M. R. O. A. Maximo, T. Yoneyama, Deep reinforcement learning for humanoid robot dribbling, in: 2020 Latin American Robotics Symposium (LARS), 2020 Brazilian Symposium on Robotics (SBR) and 2020 Workshop on Robotics in Education (WRE), IEEE, 2020, pp. 246–251.
- [43] D. P. Kingma, J. Ba, Adam: A method for stochastic optimization, CoRR 1412.6980.
- [44] J. Schulman, P. Moritz, S. Levine, M. Jordan, P. Abbeel, High-dimensional continuous control using generalized advantage estimation, CoRR 1506.02438.
- [45] H. Picado, M. Gestal, N. Lau, L. Reis, A. Tomé, Automatic generation of biped walk behavior using genetic algorithms, *Bio-inspired systems: Computational and ambient intelligence (2009)* 805–812.
- [46] N. Shafii, L. P. Reis, N. Lau, Biped walking using coronal and sagittal movements based on truncated fourier series, in: J. Ruiz-del Solar, E. Chown, P. G.

Plöger (Eds.), *RoboCup 2010: Robot Soccer World Cup XIV*, Springer Berlin Heidelberg, Berlin, Heidelberg, 2011, pp. 324–335.

- [47] H. Diedam, D. Dimitrov, P.-B. Wieber, K. Mombaur, M. Diehl, Online walking gait generation with adaptive foot positioning through linear model predictive control, in: *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems*, IEEE, 2008, pp. 1121–1126.
- [48] A. Herdt, H. Diedam, P.-B. Wieber, D. Dimitrov, K. Mombaur, M. Diehl, Online walking motion generation with automatic footstep placement, *Advanced Robotics* 24 (5-6) (2010) 719–737.
- [49] R. J. Griffin, A. Leonessa, Model predictive control for dynamic footstep adjustment using the divergent component of motion, in: *2016 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2016, pp. 1763–1768.
- [50] S. Asta, S. Sariel-Talay, Nature-inspired optimization for biped robot locomotion and gait planning, in: *European Conference on the Applications of Evolutionary Computation*, Springer, 2011, pp. 434–443.
- [51] P. MacAlpine, P. Stone, Ut austin villa: Robocup 2017 3d simulation league competition and technical challenges champions, in: *Robot World Cup*, Springer, 2017, pp. 473–485.
- [52] M. Kasaei, A. Ahmadi, N. Lau, A. Pereira, A robust model-based biped locomotion framework based on three-mass model: From planning to control, in: *2020 IEEE International Conference on Autonomous Robot Systems and Competitions (ICARSC)*, IEEE, 2020, pp. 257–262.
- [53] T. Haarnoja, A. Zhou, P. Abbeel, S. Levine, Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor, in: *International Conference on Machine Learning*, PMLR, 2018, pp. 1861–1870.