2024 European Control Conference (ECC)
June 25-28, 2024. Stockholm, Sweden

Achieving Velocity Tracking Despite Model Uncertainty for a Quadruped Robot with a PD-ILC Controller

Manuel Weiss[1,2*], Andrew Stirling[3*], Alexander Pawluchin[1], Dustin Lehmann[4], Yannis Hannemann[5], Thomas Seel[2], Ivo Boblan[1]

***Abstract*— In this study, we introduce a control strategy that combines Proportional-Derivative (PD) control with Iterative Learning Control (ILC) to enhance legged robot velocity control with only the inverse kinematics and no additional system identification. This approach leverages the real-time feedback capabilities of PD control for gait tracking while incorporating ILC's learning abilities to eliminate inaccuracies from unmodeled dynamics iteratively and to reach desired velocities without residual errors. By uniting these techniques, the proposed method empowers legged robots to adapt and optimize their control behavior, achieving and maintaining desired walking velocities. Experimental results on the physical legged robot Go1 demonstrate the effectiveness of the proposed approach, highlighting its adaptability and reliability in real-world scenarios. This research represents a first step towards overcoming high computational effort and extensive data collection for quadruped robot velocity tracking through onboard learning.**

***Index Terms*— Iterative Learning Control, Nonlinear Systems, Real-time Control, Robotics**

I. Introduction

Quadruped robots have garnered substantial attention in recent years due to their potential to navigate complex and unstructured environments with remarkable agility and adaptability [1]. This burgeoning field of robotics holds immense promise for applications ranging from search and rescue missions to planetary exploration and human assistance [2]–[5]. One critical aspect in the development and deployment of quadruped robots is the control of their legged locomotion.

Mobile robots, such as legged robots, need to track desired body velocities accurately. This is essential for, e.g., autonomous path planning. Accurately tracking desired body velocities for legged robots is difficult due to, among other things, the nonlinearity and high degrees of freedom. Model-based and machine-learning approaches have proven to be successful in achieving legged locomotion. Model-based approaches rely on a precise model of the robot and the environment. However, this modeling is technically limited and does not cover all the internal dynamics. Most model-based approaches use simplified models such as the Linear Pendulum Model (LPM) or the Single Rigid Body Model (SRB) [6], [7]. Using Zero Moment Point (ZMP) the ANYmal robot from ETH Zürich achieved dynamic locomotion by solving a convex Quadratic Programming (QP) problem [8]. The MIT mini-cheetah robot can plan desired body trajectories in real-time using model predictive control (MPC) and solving a QP problem [9]. These approaches have no guaranteed stability, and the performance depends on the convergence of the QP problem. Furthermore, the computation time for such algorithms is high, and the models do not accurately represent the real robot, even if they were not simplified. The performance of the model-based approaches is highly dependent on the accuracy of the full-body model, which often fails to capture the full dynamics of the robot.

To avoid the limitations emerging from the model-based approaches, roboticists widely embraced Machine Learning (ML) approaches. The advantages of ML locomotion approaches are that they are often lightweight and versatile. However, they usually need high computational effort. For example, the bipedal robot Cassie can perform stable walking with different gait styles at various speeds using Reinforcement Learning (RL) [10] or walk stairs [11]. The bipedal robot Digit from Agility Robotics is able to walk over rough terrain and sustain walking gaits under external force disturbances using RL [12]. Rough terrain locomotion using RL was also demonstrated for the quadrupedal robot ANYmal from ETH Zürich [13]. The drawback of all these approaches is that they are computationally expensive and require a large amount of data to train the Deep Neural Networks (DNN). Since most of the data used for DNN training is gathered from simulations, while only a small portion of data can be collected from hardware, the transfer to a real system can be challenging [14]. Therefore, model-based approaches with extensive parameter identification and data-hungry ML approaches to achieve legged locomotion would be inferior to a hybrid approach learning with little real data.

The proposed control framework leverages the advantages of Proportional Derivative (PD) control in providing real-time feedback for accurate tracking of velocity references while harnessing the learning capabilities of Iterative Learning Control (ILC) to improve control performance over time. The integration of these control techniques enables the legged robot to adapt and refine its control strategy through iterative learning, thereby enhancing its ability to achieve and maintain desired velocities. Most gait controllers need a gait library where different joint angle trajectories for different body velocities are stored. These gait libraries are usually

[*]The authors contributed equally to the work. [1]CoRoLab Berlin, Berlin University of Applied Sciences and Technology, Germany. [2]Institute of Mechatronic Systems, Leibniz University Hannover, Germany. [3]McGill University, Montreal, Canada. [4] Technische Universität Berlin, Germany. [5]HTW Berlin – University of Applied Sciences, Berlin, Germany. Funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Fundation) Project-ID 528483508 - FIP 12 and the DAAD RISE program. Corresponding author {mweiss}@bht-berlin.de

generated using a dynamics model of the robot with high computational effort. The proposed controller is able to reach desired velocities without a precalculated gait library based on the dynamics model of the robot. Therefore, the controller does not rely on heavy calculations of model dynamics, nor does it need a large amount of data and computational resources to do so. In simulation an ILC approach with model-based trajectory generation and gait library has shown promising results in tracking joint trajectories [15] but there it is not shown that the robot can actually track the velocities. Also, in contrast to the method proposed in [15] in this work, the ILC only relies on the feedforward torque of the previous step, not the total torque. Therefore, the PD-ILC proposed in this work, the PD controller generates the most torque while in [15] over time the ILC and therefore the feedforward controller is the main controller.

Our research contributes to the advancement of legged robot control techniques, paving the way for improved mobility and computational cost and enabling onboard computation without the need to simulate or compute the dynamics first. The results presented in this paper demonstrate the potential of the proposed approach for enhancing the capabilities of legged robots in various practical applications. The main contributions are:

- To the best of our knowledge, this is the first instance in which a PD-ILC controller was used to track velocities on real quadruped robot hardware.
- The proposed method works without modeled dynamics and parameter identification but in contrast to other model-free methods, with little computational cost.
- The proposed control scheme is able to learn tracking desired velocities on the hardware within seconds.

The proposed control scheme is implemented and tested on the quadrupedal robot Go1 (s. Fig. 1) from Unitree Robotics introduced in II-B. The proposed control structure is also introduced in II. The performance of the proposed controller is shown in Section III.

II. Methods

This section introduces the conventions used, the robot model, and trajectory generation. The trajectory tracking controller design and the ILC are explained in detail.

A. Conventions

All quantities in the world coordinate system have a left subscript $\mathcal{E}$, all quantities within the body coordinate system have a left subscript $\mathcal{B}$, and all quantities in the hip coordinate systems are denoted with $\mathcal{H}_l$ where $l \in \{FR, FL, RR, RL\}$. The legs l are denoted by front right (FR), front left (FL), rear right (RR), and rear left (RL).

B. Robot Model

The considered robot is a quadruped robot inspired by dog biomechanics (s. Fig. 1). Each of the four legs has three hinge joints, which are actuated by three motors, resulting in three Degrees of Freedom (DOF) for each leg. The robot has a floating base with six DOF in Cartesian space. Therefore, the robot has 18 DOF in total. The variables q_x, q_y, q_z denote the Cartesian position of the robot's trunk, while the orientation is denoted as q_ϕ, q_θ, q_ψ the z-y-x Euler angles. The floating base in the $\mathcal{B}$ frame in relation to $\mathcal{E}$ frame is denoted by

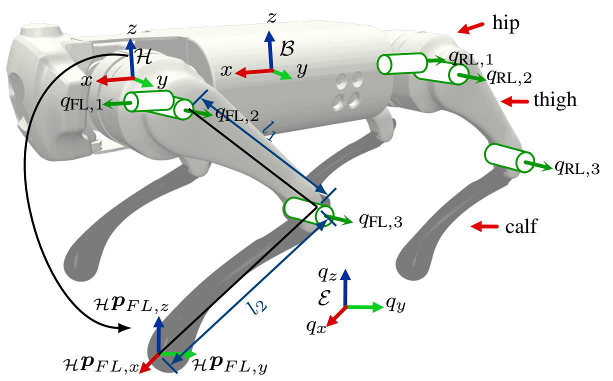

Fig. 1. Go1 kinematic model with coordinate systems and the motors of the left side.

$$ {}_{\mathcal{E}}\boldsymbol{q}_b := \left[q_x, q_y, q_z, q_\phi, q_\theta, q_\psi\right]^T . \tag{1}$$

All joints are actuated. The motor vector is given as

$$\begin{aligned}\boldsymbol{q}_m := [&q_{\mathrm{FR},1}, q_{\mathrm{FR},2}, q_{\mathrm{FR},3}, q_{\mathrm{FL},1}, q_{\mathrm{FL},2}, q_{\mathrm{FL},3},\\ &q_{\mathrm{RR},1}, q_{\mathrm{RR},2}, q_{\mathrm{RR},3}, q_{\mathrm{RL},1}, q_{\mathrm{RL},2}, q_{\mathrm{RL},3}]^T,\end{aligned} \tag{2}$$

where $q_{l,1}$ denotes the hip roll motor, $q_{l,2}$ denotes the hip pitch motor and $q_{l,3}$ denotes the knee motor for each leg l. Therefore, the generalized coordinates of the robot are denoted as

$$\boldsymbol{q} := \left[{}_{\mathcal{E}}\boldsymbol{q}_b^T, \boldsymbol{q}_m^T\right]^T . \tag{3}$$

Since the robot has four legs, there are four feet. We define the Cartesian position of these in the hip frame as:

$$ {}_{\mathcal{H}}\boldsymbol{p}_l := [p_{x_l}, p_{y_l}, p_{z_l}]^T \tag{4}$$

The foot trajectory is calculated in the hip frame to simplify the generation and can, therefore, be used by all legs. Following trajectory generation, inverse kinematics are used to calculate the joint angles and velocities for each leg. The desired joint angles can be calculated from the desired Cartesian foot position ${}_{\mathcal{H}}\boldsymbol{p}_l \in [\mathbb{P}]$ in the hip frame for the respective leg by the following inverse kinematics equations for the corresponding joint, with $\mathbb{P}$ being the set of all reachable points:

$$\begin{aligned}
q_{l,1}({}_{\mathcal{H}}\boldsymbol{p}_l) &= \arctan\left(\frac{{}_{\mathcal{H}_l}p_{y_l}}{{}_{\mathcal{H}_l}p_{z_l}}\right)\\
q_{l,2}({}_{\mathcal{H}}\boldsymbol{p}_l) &= \arccos\left(\frac{||{}_{\mathcal{H}_l}\boldsymbol{p}_l||}{l_1^2+l_2^2}\right)+\\
&\quad \arctan\left(-\frac{{}_{\mathcal{H}_l}p_{x_l}}{\sqrt{{}_{\mathcal{H}_l}p_{y_l}^2 + {}_{\mathcal{H}_l}\, p_{z_l}^2}}\right)\\
q_{l,3}({}_{\mathcal{H}}\boldsymbol{p}_l) &= \arcsin\left(-\frac{{}_{\mathcal{H}_j}p_{x_l}}{l_2} - \sin\left(q_{l,2}\right)\right) - q_{l,2}
\end{aligned} \tag{5}$$

where l_1 is the length of the thigh, and l_2 is the length of the calf. $||\cdot||$ denotes the Euler norm of the foot position vector. These inverse kinematics equations return a unique solution

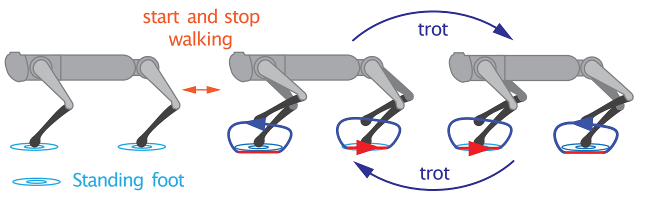

Fig. 2. Gait phases: Standing still with all feet on the ground, to start walking, the robot lifts the front left and rear right leg off the ground. The applied trotting gait alternates stand and swing phases. The diagonal feet are in phase with each other. The left feet trajectories in the hip frame are indicated in blue for the swing phase and red for the stand phase. The arrows on the feet indicate the direction of the movement.

for every reachable point. The Cartesian foot velocity v_l in the hip frame is calculated using the Jacobian (6).

$$\dot{\boldsymbol{q}}_l = \mathbf{J}_l({}_{\mathcal{H}}\boldsymbol{p}_l)\boldsymbol{v}_l \tag{6}$$

where $\mathbf{J}_l({}_{\mathcal{H}}\boldsymbol{p}_l)$ is the Jacobian matrix of the robot inverse kinematics, which is given in (5). The Jacobian matrix is defined as

$$\mathbf{J}_l({}_{\mathcal{H}}\boldsymbol{p}_l) \coloneqq \frac{\partial \boldsymbol{q}_l}{\partial_{\mathcal{H}}\boldsymbol{p}_l} \tag{7}$$

where $\boldsymbol{q}_l \coloneqq [q_{l,1}({}_{\mathcal{H}}\boldsymbol{p}_l), q_{l,2}({}_{\mathcal{H}}\boldsymbol{p}_l), q_{l,3}({}_{\mathcal{H}}\boldsymbol{p}_l)]$.

C. Trajectory Generation

There are different gait types for quadruped robots, such as trotting gait, pace gait, bounce gait, and gallop gait [16]. We propose the use of a trotting gait (trot), where the diagonal pairs of legs move in unison. Therefore, the front legs are 50 % out of phase with each other (s. Fig. 2). During the trotting gait, we split the trajectory into phases. While the stance phase describes relative movement between the foot contact point and the body, the swing phase describes the motion of the leg in the air. The given trot frequency f_t with a sampling rate f_s.

To generate the trajectories for the trotting gait, we propose using Bézier curves. These Bézier curves are used to generate the Cartesian foot position, which is subsequently used to calculate the velocity trajectory, which is defined by the start and end positions and velocities. The Bézier curve for one foot $\mathbf{B}_l(s, v_{\mathrm{com,d}}) \in \mathbb{R}^{3\times n}$ for one step s with $\frac{f_\mathrm{s}}{f_\mathrm{t}}$ normalized measured time samples, such that $s \in [0, 1]$, is defined as

$$\mathbf{B}_l(v_{\mathrm{com,d}}, s) = \sum_{i=0}^{n} \binom{n}{i} (1-s)^{n-i} s^i P_i(v_{\mathrm{com,d}}). \tag{8}$$

$s = 0$ indicates the start of the phase, and 1 the end of the current phase. The degree of the Bézier curves is defined by n, and P_i are control points of the curve for a desired linear Center of Mass (COM) velocity $v_{\mathrm{com,d}}$ along x-axis of the $\mathcal{B}$ frame. This Bézier curve is used to create the swing phase trajectory [17]. For the stand phase trajectory, a linear interpolation between the endpoint and the start point of the swing phase in the hip frame is used (indicated by the red line in Fig. 2). Since the generated swing trajectories start and end at the same distance to the ground, this should result in a stable body height. The COM velocity v_{com} depends on the distance per step; therefore, we propose not to model the dynamics of the robot but calculate the v_{com} from the distance traveled during the standing phase.

Transforming the Bézier curve $\mathbf{B}_l(v_{\mathrm{com,d}}, s)$ into the joint space is accomplished using the inverse kinematics (5 and 6) yielding the desired joint angle trajectories of all joints for a desired $v_{\mathrm{com,d}}$ which can be fully represented by the numerical matrix $\mathbf{H}_\mathrm{d}(v_{\mathrm{com,d}}, s) \in \mathbb{R}^{n_j\times n}$ and joint angular velocity trajectories $\dot{\mathbf{H}}_\mathrm{d}(v_{\mathrm{com,d}}, s) \in \mathbb{R}^{n_j\times n}$ where n_j equals the number of joints (e.g. trajectories for one leg s. Fig. 3). The desired joint space trajectories are used by the PD controller to track the desired trajectory. The trajectories are not optimized nor smoothened to be perfect in the joint space since this would require additional modeling of dynamics.

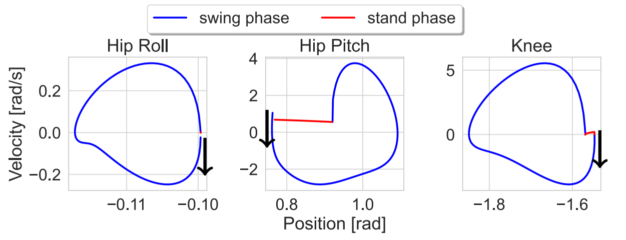

Fig. 3. Trajectories in joint space. Steps in the velocity trajectories are caused by the calculation of the position derivative. The trajectories are not smoothened nor optimized into the joint space. The black arrows indicate the start of the swing phase and the direction of the movement.

D. Trajectory Tracking Controller Design

The controller consists of a linear feedback controller and an ILC. The PD controller is used to track the desired trajectory and is solely based on the current tracking error. Each step is considered a trial. There is a trial-dependent variable s, and we use the subscript k to denote each step. In addition to the feedback term, the ILC acts as a feedforward term and learns from the errors obtained during previous trials. The ILC is used to reduce the tracking error resulting from unmodeled dynamics as well as from constant disturbances. For each of the 12 joints, an independent PD controller and ILC is designed and applied.

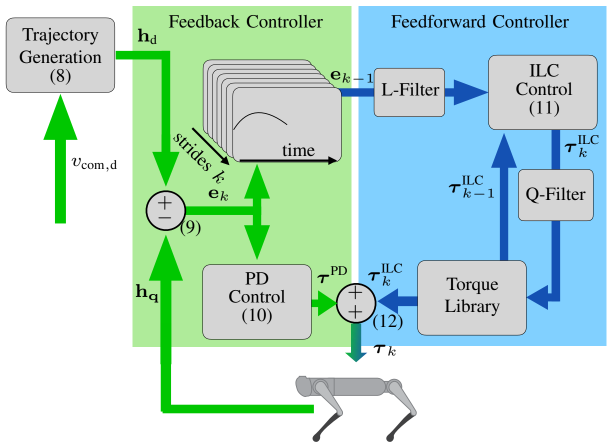

Fig. 4. Trajectory Tracking Controller Design. The detailed calculations of all elements of feedback control (green) and feedforward control (blue) are listed with their corresponding equation numbers in the paper.

1) Proportional Derivative (PD) Controller: The desired trajectories in the Cartesian space are transformed into the joint space matrices $\mathbf{H}_\mathrm{d}(v_\mathrm{com,d}, s)$ and $\dot{\mathbf{H}}_\mathrm{d}(v_\mathrm{com,d}, s)$ where each row represents the trail trajectory for one joint and each column represents all desired joint angles and angular velocities respectively at a normalized time step s as described in the previous section. The current desired angles and angular velocities are denoted as $\mathbf{h}_\mathrm{d}(s) \in \mathbb{R}^{n_j}$, $\dot{\mathbf{h}}_\mathrm{d}(s) \in \mathbb{R}^{n_j}$. The desired trajectories are the input for the PD feedback controller.

Since the trajectories are generated in Cartesian space and not optimized nor smoothened, the velocity trajectories are not perfect, hence the steps in the trajectories (s. Fig. 3). The joint velocities are continuous within each phase; during the phase change, the trajectory is discontinuous, which will lead to some minor errors since the sudden step in the velocity is not possible for the motor. The position and velocity error are calculated as follows:

$$\begin{aligned} \mathbf{e}_k(s) &= \mathbf{h}_\mathrm{d}(s) - \mathbf{h}_\mathbf{q}(s) \\ \dot{\mathbf{e}}_k(s) &= \dot{\mathbf{h}}_\mathrm{d}(s) - \dot{\mathbf{h}}_\mathbf{q}(s) \end{aligned} \tag{9}$$

where $\mathbf{h}_\mathbf{q}(s)$ are the measured joint angles and $\dot{\mathbf{h}}_\mathbf{q}(s)$ the measured joint angular velocities respectively. The output of the PD controller $\boldsymbol{\tau}^\mathrm{PD}(s) \in \mathbb{R}^{12}$ is calculated as follows:

$$\boldsymbol{\tau}^\mathrm{PD}(s) = \mathbf{K}_\mathrm{P}\mathbf{e}_k(s) + \mathbf{K}_\mathrm{D}\dot{\mathbf{e}}_k(s) \tag{10}$$

where $\mathbf{K}_\mathrm{P} \in \mathbb{R}^{n_j \times n_j}$ and $\mathbf{K}_\mathrm{D} \in \mathbb{R}^{n_j \times n_j}$ are the proportional and derivative gains, respectively. To avoid a sim-to-real gap and needing as little model information as possible, the gains $\mathbf{K}_\mathrm{P}$ and $\mathbf{K}_\mathrm{D}$ were tuned by suspending the robot and tracing the trajectory with all feet in the air, which is possible due to the robot's lightweight. For simplification, the gains are uniform for all joints ($\mathbf{K}_\mathrm{P} = \mathbf{I}_{12} \cdot k_\mathrm{P}$ and $\mathbf{K}_\mathrm{D} = \mathbf{I}_{12} \cdot k_\mathrm{D}$ where $\mathbf{I}$ denotes the identity matrix). This can be done since the motors for all joints are similar, although all motors experience different forces due to the weight distribution on the torso. Therefore, tuning the k_P and k_D for each motor independently would require knowledge of the dynamics and inertia of the robot.

2) Iterative Learning Control (ILC) Controller: The quadruped robot is a highly dynamic and nonlinear system where dynamic forces are modeled to generate the trajectory or to design a feedforward controller [18]. To correct the gait for the unmodeled dynamics and constant disturbances, we propose to use an ILC to minimize the joint position error. Since the ILC minimizes the joint angular position error and angular velocity error, the desired joint trajectories are followed; therefore, the desired body velocity $v_\mathrm{com,d}$ is reached. Since all gaits, like the trot, are based on a repeating sequence of trajectories, the ILC as a model-free approach is suited to improve the trajectory tracking [19]. To update the feedforward torque, the trajectory error in position and velocity from the previous stride are used. Due to the different dynamics of each leg, 12 independent ILCs are used. Due to the delay in the hardware, the time data is shifted $0.01\,\mathrm{s}$ ahead (δ) to improve the ILC's performance. To update the feedforward torque we use the same $\mathbf{K}_\mathrm{P}$ and $\mathbf{K}_\mathrm{D}$ gains as in the PD-Controller:

$$\begin{aligned} \boldsymbol{\tau}_k^\mathrm{ILC}(s) = \boldsymbol{\tau}_{k-1}^\mathrm{ILC}(s) &+ l_\mathrm{P} \cdot \mathbf{K}_\mathrm{P}\mathbf{e}_{k-1}(s+\delta) \\ &+ l_\mathrm{D} \cdot \mathbf{K}_\mathrm{D}\dot{\mathbf{e}}_{k-1}(s+\delta) \end{aligned} \tag{11}$$

where $l_\mathrm{P} = 0.2$, $l_\mathrm{D} = 0.1$ are the ILC gains. Therefore, the ILC compensates $20\,\%$ of the remaining position errors and $10\,\%$ of the remaining velocity errors. The total torque is then calculated by adding the feedback torque and the feedforward torque:

$$\boldsymbol{\tau}_k(s) = \boldsymbol{\tau}^\mathrm{PD}(s) + \boldsymbol{\tau}_k^\mathrm{ILC}(s) \tag{12}$$

where $\boldsymbol{\tau}(s) \in \mathbb{R}^{12}$ contains the torque for all joints. The ILC is always updated after the stand phase therefore, the front right and the front left leg are updated asynchronously with a phase shift of 0.5. This ensures that all trajectories are processed in the same way. The ILC torque $\boldsymbol{\tau}_k^\mathrm{ILC}$ is stored in the torque library therefore $\boldsymbol{\tau}_{k-1}^\mathrm{ILC}$ can be used in the ILC update and after the ILC has converged. To smoothen the ILC inputs, all errors are pre-filtered (L-Filter) using a 4th-order Infinite Impulse Response (IIR) low pass filter, which is applied through a forward and backward pass. An identical filter is applied in the same fashion to post-filter (Q-Filter) the torques for the same reason after the update. Additionally, we filter the measured COM velocity v_com using a running mean over the last second.

III. Results

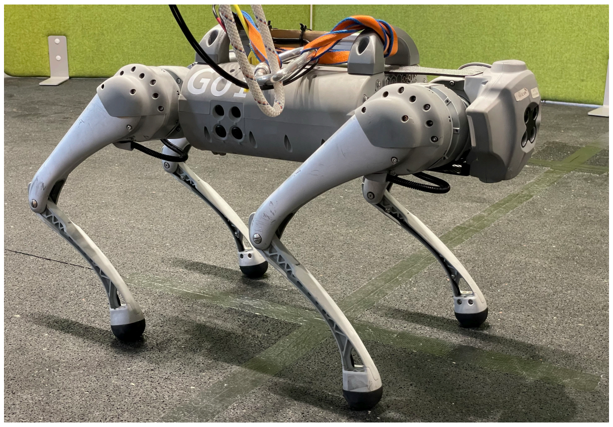
Fig. 5. The Go1 experimental setup. The Robot is tethered for safety and to separate power and Ethernet cables.

The proposed control strategy is tested on the Unitree Go1 Edu. Go1 weighs approximately $12\,\mathrm{kg}$ with most of the mass located in the torso. To test the controller, a desired COM velocity $v_\mathrm{com,d}$ is set, and the robot starts from standing still. For safety reasons and to hold the power and the Ethernet cable away from the legs of the robot, the Go1 is connected to a safety rope (s. Fig 5). The communication between the operating computer and the robot is implemented in Robot Operating System (ROS) [20]. The step duration is set to $0.5\,\mathrm{s}$ resulting in 2 full step cycles per leg per second and the controller runs at $500\,\mathrm{Hz}$. Changes in velocity are achieved by changing the distance per step. Since the robot starts from standing still, the first steps are needed to accelerate the joints. Therefore, the ILC starts after the third step is taken. Before that, the joints have not accelerated enough to make reasonable corrections.

The ILC is updated until the Root Mean Square (RMS) of $\mathbf{e}_k(s) \approx 0$ or until the tenth step. The error in velocity $\mathbf{e}_{\text{vel}}$ is neglected in this condition, given that the velocity is the first derivative of the position – the velocity error is minimal. The ten-step limit for the ILC was chosen only to have to walk at the desired speed for $5\,\mathrm{s}$ until the torque is stored for later use in a torque library. Using these stored torques as feedforward torques, the robot can reach the velocities without updating using the ILC-controller. This allows it to learn every COM velocity once and use that velocity from then on.

To test the controller different desired COM velocities $v_{\text{com,d}} \in [-0.2, -0.1, 0.1, 0.3, 0.4]\,\mathrm{m\,s^{-1}}$ were used. First, all velocities are tested using only the PD-controller, consisting of 12 Single Input Single Output (SISO) PD controllers. With the PD baseline, the ILC is tested. The error while using solely the PD controller is above $20\,\%$ this is due to the unmodeled dynamics, constant disturbances, and the imperfect PD gains, which we set to $k_{\text{P}} = 90$ and $k_{\text{D}} = 4$ (PD tuning described above). About three full step cycles are needed to reach a stable COM velocity.

The ILC correction can be observed in the trajectories (s. Fig. 6). It is clearly visible that the ILC is able to compensate for the unmodeled dynamics and correct the motor position. The initial error is always more significant on the rear legs. The hip motors should perform only small motions, which is not the case initially. During the stand phase, the desired hip position does not change, but due to gravity and other unmodeled dynamics, the hip motor does not hold the desired position. The ILC improves the performance, but there is still some movement in the joint. However, since the error on the hip motor never exceeds $0.0262\,\mathrm{rad}$ ($\approx 1.5\,\mathrm{deg}$) after the convergence of the ILC, this does not impede the performance. Unmodeled dynamics have little impact on the thigh joints from the beginning. Therefore, the controller complexity could be reduced for this case by removing the ILC for these joints. Yet, the rear legs improve in tracking the trajectory. The calf joints are most impacted by unmodeled dynamics. Here, the gravity does impact the stand phase trajectory and makes tracking impossible without the ILC.

During the swing phase, the PD controller is able to follow the trajectory. This is expected since the PD controller was tuned to track the trajectory in the air. During the stand phase, gravity impacts the rear legs more than the front legs due to the robot's mass distribution. Therefore, to track the trajectory without the ILC using only a PD controller, some modeling would be required. As all PD controllers are equivalent, gravity exerts a greater influence on the rear legs, resulting in a more pronounced ILC correction for the rear legs. After the seven ILC update steps, the trajectory tracking is not perfect but satisfactory in reaching the desired COM velocity $v_{\text{com,d}}$. The robot's rear legs are not yet at the same height with regard to the hip. Therefore, the robot's torso leans backward with a pitch angle below $5\,\mathrm{deg}$; this is visible in Fig. 6 due to the higher calf angle error during the stance phase.

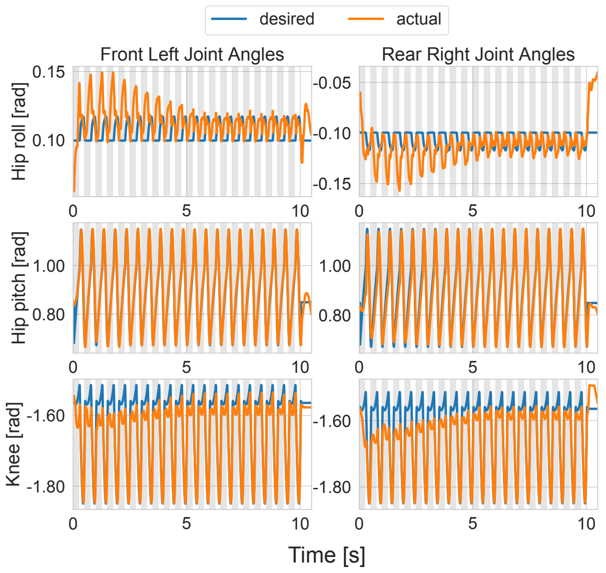

Fig. 6. Improvement over time in tracked position using ILC for $0.4\,\mathrm{m/s}$. The ILC correction is evident on the calf joints, where the most significant improvement occurs. The grey shaded areas indicate the stand phase.

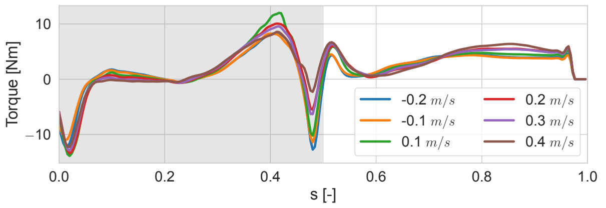

Fig. 7. Learned ILC torques for the different COM velocities (Front Right calf joint). The grey shaded area indicates the stand phase.

Although the ILC does mostly improve the trajectory during the stance phase, the ILC does not primarily compensate for gravity (s. Fig. 7). During a great part of the stance phase $\boldsymbol{\tau}^{\text{ILC}} \approx 0$. Also, the feedforward torque is different for different $v_{\text{com,d}}$ especially in the end of the stance phase. Therefore, the PD controller can compensate for most of the gravity, and the ILC compensates for unmodeled dynamics and the remaining error resulting from gravity.

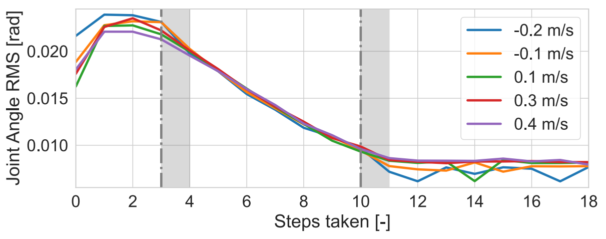

Fig. 8. Position RMS error for different COM velocities $v_{\text{com,d}}$ in each ILC trial. The grey vertical lines represent the initial and final ILC-update. The grey shaded areas indicate the delay until all torques are updated and applied once (i.e., the error minimizes further after the last update). The initial errors for all velocities are similar and are corrected at the same rate.

The PD-IL controller successfully compensates for the unmodeled dynamics and reaches $v_{\mathrm{com,d}}$ with a remaining error below $5\,\%$ (s. Fig. 9). The final error arises from slip between the feet and the ground, which cannot be compensated using only the feet trajectories relative to the $\mathcal{B}$ or $\mathcal{H}$ frames. This could be avoided by improving the trajectory generation to consider friction constraints and generate a more realistic trajectories during the stance phase. The ILC minimizes the position error as shown in Fig. 8 where the error in position is calculated as the RMS error of the position over the whole trajectory for all joints. Here, the convergence of the ILC is clearly visible. Regardless of speed, the error at the beginning and the end of the ILC updates matches closely. After the last update, the error is invariant, which explains the stable COM velocities $v_{\mathrm{com,d}}$ – the feedforward torque is always applied. Walking backward is particularly challenging for the PD controller. This is due to the PD controller's inability to compensate gravity, unmodeled dynamics on the rear legs, causing the front legs to push the rear feet further into the ground. The performance improvements after the last ILC update can be attributed to the new torque which is first applied after the final update and the inertia of the robot.

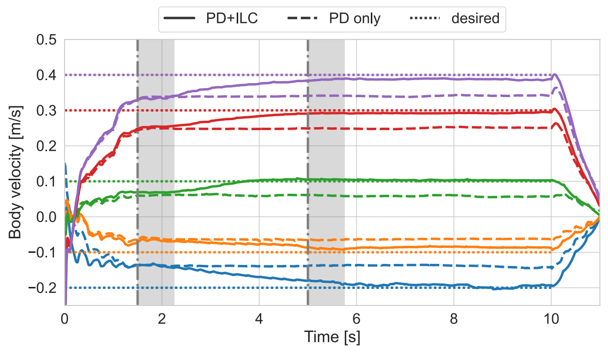

Fig. 9. COM velocities reached with and without ILC for five arbitrarily chosen references. It is visible that with the ILC, the performance is significantly better, and it is possible to track the desired velocities. The grey vertical lines represent the initial and final ILC-update. The grey shaded areas indicate the delay until all torques are updated and applied once (i.e., the error minimizes further after the last update).

Since there is no feedback for the velocity, the proposed method is only valid as long as there is little to no slip between the feet and the ground. The Unitree Go1, equipped with rubber feet, allows for this assumption across various surfaces. However, the rapid seven-step ILC update process, despite its efficiency, requires time to learn the torques for different speeds.

IV. Conclusion

In this paper, we proposed and tested a control scheme using a PD-ILC to achieve desired velocities in quadruped robots. The proposed control framework leverages the advantages of PD control in providing real time feedback for accurate tracking of velocity references while harnessing the learning capabilities of ILC to improve control performance over time. The integration of these control techniques enables the legged robot to adapt and refine its control strategy through iterative learning, thereby enhancing its ability to achieve and maintain desired velocities. With the proposed control scheme, we rapidly learn unmodeled dynamics and are able to achieve accurate trajectory tracking without relying on heavy calculations or large amounts of data gathered in simulation or on the robot. Since all data is gathered in real time on the robot, there is no sim-to-real gap.

This paper shows a way to overcome high modeling and PD controller tuning effort as well. The proposed control scheme is an important step toward overcoming high computation costs and achieving on-board gait generation and control. It might be possible to get similar results with a perfectly tuned Proportional Integral Derivative (PID) controller, but that would require more modeling and optimization of the controller gains, which can be very difficult or time-consuming. Testing this method on additional quadruped robots is necessary to demonstrate its efficacy across various kinematic and dynamic configurations, as this aspect remains unverified in the present paper, thereby delineating the scope for future investigation.

In future research, we plan to teach the robot various gait patterns to assess the performance of ILC with diverse movements. With different gait patterns or higher step frequencies, higher speeds would also be possible. In forthcoming research, the prediction of the acquired ILC torques through Gaussian Process Regression will be undertaken.

References

[1] L. Wellhausen and M. Hutter, "Rough terrain navigation for legged robots using reachability planning and template learning," in *2021 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2021, pp. 6914–6921.

[2] M. Hutter, C. Gehring, D. Jud, A. Lauber, C. D. Bellicoso, V. Tsounis, J. Hwangbo, K. Bodie, P. Fankhauser, M. Bloesch, R. Diethelm, S. Bachmann, A. Melzer, and M. Hoepflinger, "Anymal - a highly mobile and dynamic quadrupedal robot," in *2016 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2016, pp. 38–44.

[3] N. Li, J. Cao, and Y. Huang, "Fabrication and testing of the rescue quadruped robot for post-disaster search and rescue operations," in *2023 IEEE 3rd Int. Conf. on Electronic Technology, Communication and Information (ICETCI)*, 2023, pp. 723–729.

[4] C. Cruz Ulloa, J. del Cerro, and A. Barrientos, "Mixed-reality for quadruped-robotic guidance in SAR tasks," *Journal of Computational Design and Engineering*, vol. 10, no. 4, pp. 1479–1489, 06 2023. [Online]. Available: https://doi.org/10.1093/jcde/qwad061

[5] A. Swaminathan, S. R, J. B. J, and J. V, "Design and development of light weight and low-cost quadruped robot for spying and surveillance," in *2022 Int. Conf. on Innovation and Intelligence for Informatics, Computing, and Technologies (3ICT)*, 2022, pp. 500–504.

[6] S. Kajita, F. Kanehiro, K. Kaneko, K. Yokoi, and H. Hirukawa, "The 3d linear inverted pendulum mode: a simple modeling for a biped walking pattern generation," in *Proceedings 2001 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems. Expanding the Societal Role of Robotics in the the Next Millennium (Cat. No.01CH37180)*, vol. 1, 2001, pp. 239–246 vol.1.

[7] B. Katz, J. D. Carlo, and S. Kim, "Mini cheetah: A platform for pushing the limits of dynamic quadruped control," in *2019 Int. Conf. on Robotics and Automation (ICRA)*, 2019, pp. 6295–6301.

[8] C. Dario Bellicoso, F. Jenelten, P. Fankhauser, C. Gehring, J. Hwangbo, and M. Hutter, "Dynamic locomotion and whole-body control for quadrupedal robots," in *2017 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2017, pp. 3359–3365.

[9] D. Kim, J. D. Carlo, B. Katz, G. Bledt, and S. Kim, "Highly dynamic quadruped locomotion via whole-body impulse control and model predictive control," *CoRR*, vol. abs/1909.06586, 2019. [Online]. Available: http://arxiv.org/abs/1909.06586

[10] Z. Xie, P. Clary, J. Dao, P. Morais, J. Hurst, and M. van de Panne, "Iterative reinforcement learning based design of dynamic locomotion skills for cassie." [Online]. Available: http://arxiv.org/abs/1903.09537

[11] J. Siekmann, K. Green, J. Warila, A. Fern, and J. W. Hurst, "Blind bipedal stair traversal via sim-to-real reinforcement learning," *CoRR*, vol. abs/2105.08328, 2021. [Online]. Available: https://arxiv.org/abs/2105.08328

[12] G. A. Castillo, B. Weng, W. Zhang, and A. Hereid, "Robust feedback motion policy design using reinforcement learning on a 3d digit bipedal robot," in *2021 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, pp. 5136–5143, ISSN: 2153-0866.

[13] J. Lee, J. Hwangbo, L. Wellhausen, V. Koltun, and M. Hutter, "Learning quadrupedal locomotion over challenging terrain," *Science Robotics*, vol. 5, no. 47, Oct 2020. [Online]. Available: https://doi.org/10.1126%2Fscirobotics.abc5986

[14] W. Zhao, J. P. Queralta, and T. Westerlund, "Sim-to-real transfer in deep reinforcement learning for robotics: A survey," *CoRR*, vol. abs/2009.13303, 2020. [Online]. Available: https://arxiv.org/abs/2009.13303

[15] J. Cheng, Y. G. Alqaham, A. K. Sanyal, and Z. Gan, "Practice Makes Perfect: An iterative approach to achieve precise tracking for legged robots." [Online]. Available: http://arxiv.org/abs/2211.11922

[16] Y. Fukuoka and H. Kimura, "Dynamic locomotion of a biomorphic quadruped 'tekken' robot using various gaits: walk, trot, free-gait and bound," vol. 6, no. 1, pp. 63–71. [Online]. Available: http://content.iospress.com/doi/10.1080/11762320902734208

[17] D. J. Hyun, S. Seok, J. Lee, and S. Kim, "High speed trot-running: Implementation of a hierarchical controller using proprioceptive impedance control on the MIT Cheetah," 2014. [Online]. Available: https://dspace.mit.edu/handle/1721.1/98270

[18] P. Corke, W. Jachimczyk, and R. Pillat, "Dynamics and control," in *Robotics, Vision and Control: Fundamental Algorithms in MATLAB®*. Springer Int. Publishing, pp. 355–392. [Online]. Available: https://doi.org/10.1007/978-3-031-07262-8_9

[19] S. L. Brunton and J. N. Kutz, *Data-Driven Science and Engineering: Machine Learning, Dynamical Systems, and Control*. Cambridge University Press, 2019.

[20] Stanford Artificial Intelligence Laboratory et al., "Robotic operating system." [Online]. Available: https://www.ros.org