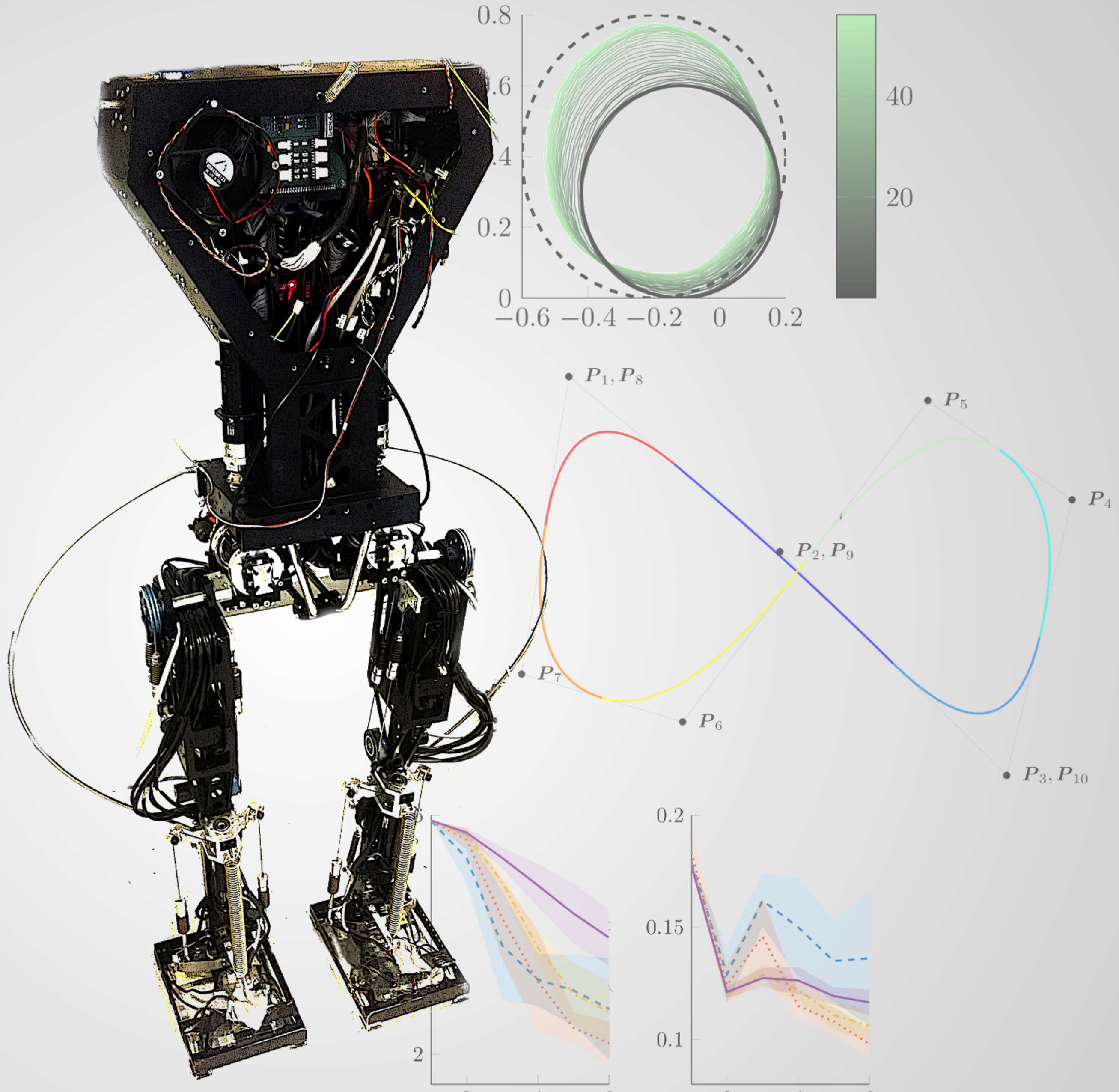
Towards Incremental Kinesthetic Teaching of Bipedal Walking

M. van Lohuijzen



TUDelft

Towards Incremental Kinesthetic Teaching of Bipedal Walking

by

M. van Lohuijzen

to obtain the degree of Master of Science
at the Delft University of Technology,
to be defended publicly on Friday September 8, 2017 at 14:00.

Student number:	4018001	
Project duration:	Oktober 1, 2016 – September 8, 2017	
Thesis committee:	Dr. ir. H. Vallery,	TU Delft, supervisor
	Prof. dr. ir. M. Wisse,	TU Delft
	Ir. I. Koryakovskiy,	TU Delft

Towards incremental kinesthetic teaching of bipedal walking

Michiel van Lohuijzen
Delft University of Technology
Email: m.vanlohuijzen@student.tudelft.nl

***Abstract*—The large dimensionality of walking motions is a challenge for robot learning. The human seems designated to assist in this learning process, because of their aptness in walking. This paper presents a step in the investigation how a human can teach a robot a walking-like motion using incremental kinesthetic teaching. This approach lets the human evaluate and correct the teaching actions during robot learning. A state-dependent tracking method is designed, which allows for spatio-temporal variations of the trajectory during the teaching process. A model-free iterative learning control method identifies a torque trajectory for accurate reference tracking even with low-impedance feedback control. The human teacher switches between iterative learning control and incremental kinesthetic demonstrations with a button press. To investigate the teaching performance of the human, a metric is introduced representing the error between a predefined target trajectory and the reference trajectory as taught to the robot. Experiments with one leg of the TUlip humanoid robot show accurate tracking performance of the iterative learning controller. They identify an optimal learning rate of the incremental kinesthetic teaching algorithm with respect to the teaching performance of a human subject. However, unintuitive indication of demonstration periods decreases the teaching performance, such that a significant error between the target and the taught reference trajectory still exists. Future work should focus on a more intuitive interface to teach whole body motions more accurately.**

I. Introduction

Robust, energy-efficient and fast bipedal walking forms a key challenge in robotics. This is because it is an optimal way of locomotion in a human-shaped world, it provides us with insight into human motor control and it combines important difficulties in control engineering: floating-base non-linear hybrid dynamics, under-actuation and the fact that the desired behavior is not a steady state, but a limit-cycle motion.

To find such a limit cycle and create a policy for bipedal walking robots that is robust against model uncertainties and disturbances, robot learning is required because it does not rely on an accurate model and is less complex to generalize.

There are multiple problems with robot learning for bipedal walking. The high dimensionality of the configuration space and control space of the robot leads to a large search space [1]. This makes it difficult to find an optimal policy. Exploration must be performed cautiously to prevent falling, as it can damage the robot. Robot learning from demonstration (RLfD) [2] lets a human assist the machine learning process to solve this problem. In RLfD a human teaches a robot a movement by demonstrating it, instead of programming [3]. Although the human is not an expert in robot motions, relevant parts of the state space can be indicated and alternatives can be suggested to aid the exploration.

Michiel van Lohuijzen was with the Department of Mechanical Engineering, Delft University of Technology, Delft

There are many examples in literature that use a batch of motion demonstrations from a human teacher to initialize the learning process. (e.g. [4], [5], [6]). However, human-robot interaction stops after initialization and the robot is left on its own for the rest of the learning process. Allowing the teacher to evaluate the behavior of the robot as it learns from the demonstrated motions and adapt the demonstrations incrementally based on the findings, can speed up the learning process [7]. Examples of literature that follow this approach are [8] and [9].The work of [8] uses motion capture and [9] uses kinesthetic teaching, where the joints of the robot are passive as the human demonstrates the motion. However, the latter neglects the advantage of kinesthetic teaching that the human uses haptic feedback to evaluate the robot motion. Moreover, letting the robot actively execute the motion during the incremental learning process simplifies the teaching task. The human can make small adjustments to faulty parts of the trajectory specifically and does not need to demonstrate the entire trajectory during each trial [10].

This paper presents a step in the investigation of incremental kinesthetic teaching (IKT) by a human, using impedance control with the objective to teach a robot a walking motion. The research is divided in two parts. The first part, presented in this work, considers how IKT with impedance control can be used, such that a human can teach a desired joint motion to the robot. The second part should answer if the human is able to identify which trajectory leads to limit cycle walking.

The presented approach overcomes problems of the application of existing IKT methods [11] and [12] to teaching walking motions. The methods use model-based impedance control and a motion refinement tube around the trajectory with locally high impedance that the human needs to overcome to indicate demonstrations. Like most RLfD approaches, the motion is encoded with motion primitives, using a hidden Markov model (HMM) [13]. The works of [11] and [12] use a forgetting factor, which defines the learning rate, i.e. how much each new demonstration influences the trajectory. This proposes the following problems:

- The joint impedance of the robot used in this work is limited due to the use of series elastic actuation [14] to allow for energy efficiency [15]
- The learning method and controller must be model-free.

- There is no velocity reference tracking available on the used robot.
- HMMs are discrete models that need explicit transition smoothing [11]. Creating a limit cycle walking motion prefers the use of an implicitly continuous trajectory representation.

Moreover, learning performance is not assessed quantitatively in literature with respect to this forgetting factor.

To overcome these problems, this work proposes a B-spline representation [16] to encode the desired trajectory. The B-spline representation is used because of its attractive features like continuity, an easily computed derivative, local controllability and the availability of linear regression fitting method [17]. The closest point on the trajectory with respect to the measured robot position is used as control reference, which renders a state-dependent reference. A damped least squares algorithm [18] updates the trajectory at a predefined measurement rate. A model-free interaction controller is designed. It consists of a torque trajectory and a low-impedance feedback controller, allowing interaction with a human. The torque trajectory is trained using iterative learning control (ILC) [19]. This is motivated by its simple implementation and that it was found in [20] to be very effective for the cyclic task of walking, with human interaction.

In an experiment on the TUlip humanoid robot it is shown that the proposed method allows IKT between the human and the robot. A target motion is given that a human subject needs to teach to the robot. A performance measure is defined as the error between the target and taught reference motion. This allows for quantitative assessment of the teaching performance. Performing multiple trials with different learning rates shows an optimal rate with respect to the learning performance. Moreover, this work shows that ILC is a viable model-free approach to create accurate tracking, in spite of the hardware limits and low-impedance feedback control.

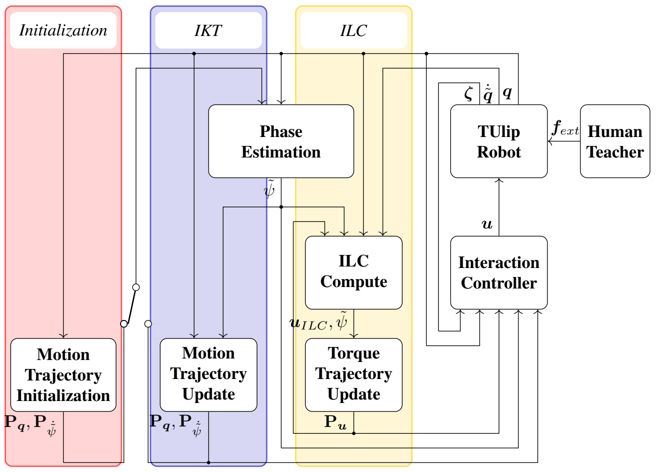
The paper is structured into the following sections: First, Section II provides an overview of the complete teaching scheme. It is succeeded by Section III, which discusses the use of B-splines as trajectory representation and the initial trajectory fit. Section IV presents how the closest point on the trajectory is estimated. Section V then presents the interaction controller. Section VI discusses the algorithms used to update the trajectories. Section VII presents the experimental setup and Section VIII shows their results. Section IX presents a discussion of the results. Section X proposes future research directions and Section XI concludes this work.

II. Incremental learning scheme

This section provides an overview of the motion initialization, ILC and IKT scheme.

A. *Initializing the teaching and iterative learning control processes*

The scheme of the total process is provided in figure 1. The process starts with the *Initialization* after a button press from the teacher. The **Interaction Controller** block is turned off, such that the robot is passive and the teacher can demonstrate



Initialization
IKT
ILC
Phase Estimation
$\tilde{\psi}$
TUlip Robot
$\boldsymbol{f}_{ext}$
Human Teacher
$\boldsymbol{\zeta}$ $\dot{\boldsymbol{q}}$ $\boldsymbol{q}$
$\boldsymbol{u}$
ILC Compute
Interaction Controller
$\boldsymbol{u}_{ILC}, \tilde{\psi}$
Motion Trajectory Initialization
Motion Trajectory Update
Torque Trajectory Update
$\mathbf{P}_{\boldsymbol{q}}, \mathbf{P}_{\dot{\psi}}$
$\mathbf{P}_{\boldsymbol{q}}, \mathbf{P}_{\dot{\psi}}$
$\mathbf{P}_{\boldsymbol{u}}$

Fig. 1: Scheme of the IKT system. The outputs of the robot $\boldsymbol{q}$, $\dot{\boldsymbol{q}}$ and $\boldsymbol{\zeta}$ indicate the joint positions, joint velocity estimations and motor velocities. Control points $\mathbf{P}_{\boldsymbol{q}}$, $\mathbf{P}_{\psi}$ and $\mathbf{P}_{\boldsymbol{u}}$ are the parameters of the joint, phase and torque trajectory respectively. The vectors $\boldsymbol{u}$ and $\boldsymbol{u}_{ILC}$ are the joint and iterative learning control torques, the value $\tilde{\psi}$ is the phase estimation and $\boldsymbol{f}_{ext}$ is the force as applied by the human teacher.

an initial motion. At each measurement instance k, the robot joint positions are saved in **Motion Trajectory Initialization**. After a second button press, this block fits a B-spline motion trajectory, that is a combination of a position trajectory and a velocity magnitude trajectory.

The initial fit activates the *ILC* loop and the **Interaction Controller**. A button press by the human teacher then switches between the *ILC* and *IKT* mode.

B. *The teaching and iterative learning control processes*

At each time step t_s the **Phase Estimation** block estimates a location (hereafter indicated as the phase) on the trajectory that is the closest to the current joint configuration of the robot. A (Gauss-)Newton algorithm is used to estimate this point.

The **Interaction Controller** is designed to track the motion references while allowing interaction with a human, in spite of the lack of accurate velocity measurements in the joints. It is a combination of a torque trajectory and a proportional derivative (PD) feedback controller. Motion demonstrations by the human are possible due to the low impedance of the feedback controller. The torque trajectory is encoded in the B-spline framework, together with the motion trajectory. The phase, estimated in the **Phase Estimation** block, defines the motion and torque references to allow the human to slow down or speed up the motion.

In *ILC* mode the torque trajectory is trained each measurement instance t_k. The **ILC Compute** block uses the phase estimate to indicate the motion reference and compute the ILC torques. The **Torque Trajectory Update** block updates the torque trajectory directly each time instance t_k using the ILC torques. The phase estimate defines local weights of the torque data on the B-spline parameters.

When the system is in *IKT* mode the human pushes the robot limbs to indicate a change in the trajectory. The **Motion Trajectory Update** block measures the positions at each measurement instance t_k and updates the motion trajectory parameters directly using a damped least squares algorithm and the local weights, computed using the phase estimate.

III. Trajectory initialization

This section presents how the motion trajectory is encoded and how it is initialized.

A. Trajectory representation

A cubic (degree $o = 3$) B-spline [21] representation is used for the trajectory. (See Appendix A for more information on B-splines.) This section describes how the robot motion and torque trajectories are encoded as a B-spline.

Given that the joint configuration of the robot is indicated by $\boldsymbol{q}(t) \in \mathbb{R}^n$ at time t, consider B-spline trajectories:

$$\begin{bmatrix} \boldsymbol{q}_{\text{ref}}(\psi) \\ \dot{\psi}_{\text{ref}}(\psi) \\ \boldsymbol{u}_{\text{ref}}(\psi) \end{bmatrix} = \begin{bmatrix} \mathbf{P}_{\boldsymbol{q}} \\ \mathbf{P}_{\dot{\psi}} \\ \mathbf{P}_{\boldsymbol{u}} \end{bmatrix} \boldsymbol{w}(\psi). \tag{1}$$

Here ψ is the phase. The function $\boldsymbol{q}_{ref}(\psi) \in \mathbb{R}^n$ is the joint position trajectory, $\dot{\psi}_{\text{ref}}(\psi) \in \mathbb{R}$ is the reference of the velocity magnitude along the trajectory and $\boldsymbol{u}_{\text{ref}}(\psi) \in \mathbb{R}^n$ is the torque reference trajectory.

The matrix $\mathbf{P}_{\boldsymbol{q}} \in \mathbb{R}^{n\times m}$ contains the m control points, which define the shape of the n-dimensional joint position trajectory. The matrix $\mathbf{P}_{\dot{\psi}} \in \mathbb{R}^{1\times m}$ consists of the control points, which are the parameters of $\dot{\psi}_{\text{ref}}(\psi)$ and $\mathbf{P}_{\boldsymbol{u}} \in \mathbb{R}^{n\times m}$ contains the control points of $\boldsymbol{u}_{\text{ref}}(\psi)$. The vector $\boldsymbol{w}(\psi) \in \mathbb{R}^m$ contains the basis functions computed using the Cox-de Boor recursion formula [16], which define the local weights of the control points on the trajectory.

The joint velocity reference trajectory $\dot{\boldsymbol{q}}_{\text{ref}}(\psi)$ can be computed knowing the trajectories $\boldsymbol{q}_{\text{ref}}(\psi)$ and $\dot{\psi}_{\text{ref}}(\psi)$ as in Equation 2.

$$\dot{\boldsymbol{q}}_{\text{ref}}(\psi) = \boldsymbol{q}'_{\text{ref}}(\psi)\dot{\psi}_{\text{ref}}(\psi), \tag{2}$$

where $\boldsymbol{q}'_{\text{ref}}(\psi)$ is the derivative of $\boldsymbol{q}_{\text{ref}}(\psi)$ to ψ, as computed as in Equation 28 of Appendix A.

B. Initial trajectory fit

Let $\mathbf{M}_{\boldsymbol{q}} \in \mathbb{R}^{N_{\text{init}}\times n}$ be the data matrix of the measured joint positions $\boldsymbol{q}(t_k)^T$ at time t_k with a row for each measurement $k \in \{1, \dots, N_{\text{init}}\}$. N_{init} is the amount of measurements used during initialization. A fixed-knot least squares approach [17] is used to fit the spline $\boldsymbol{q}_{\text{ref}}(\psi)$ on $\mathbf{M}_q$. Upper and lower phase boundaries Ψ_u and Ψ_l are defined arbitrarily at 0 and 2π respectively. A phase location $\psi_k \in [\Psi_l, \Psi_u]$ is assigned to each measurement, such that these measurements are evenly spread over the spline. A weight vector $\boldsymbol{w}(\psi_k)$ is computed for each measurement use the Cox-de Boor recursion formula [16]. These values fill the matrix as in Equation 3.

$$\mathbf{W} = \begin{bmatrix} \boldsymbol{w}^T(\psi_1) \\ \boldsymbol{w}^T(\psi_2) \\ \vdots \\ \boldsymbol{w}^T(\psi_{N_{\text{init}}}) \end{bmatrix}. \tag{3}$$

Following Appendix A-B, the matrix $\mathbf{P}_{\boldsymbol{q}}$ is estimated as in Equation 4 with lagrange multipliers $\mathbf{\Lambda}$ [22] and the constraint matrix as in Equation 5.

$$\begin{bmatrix} \mathbf{P}_{\boldsymbol{q}} \\ \mathbf{\Lambda} \end{bmatrix} = \begin{bmatrix} \mathbf{M}_{\boldsymbol{q}}^T\mathbf{W} \\ \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{W}^T\mathbf{W} & \mathbf{T}^T \\ \mathbf{T} & \mathbf{0} \end{bmatrix}^{-1} \tag{4}$$

and

$$\mathbf{T}_o = \begin{bmatrix} \mathbf{I}_o & \mathbf{0}_{o\times m-2o} & -\mathbf{I}_o \end{bmatrix}, \tag{5}$$

where $\mathbf{I}_o$ represents an $o \times o$ identity matrix and order $o = 3$. The constraints make sure that the estimated spline is a closed loop.

IV. Phase estimation

This section discusses the phase and velocity magnitude estimations.

A. Phase position estimation

The closest point on the trajectory with respect to the current joint configuration is based on the Euclidean distance. Because the joint space is not Euclidean, there is no trivial measurement for distance and the closest point is not clearly defined [23]. However, for small errors in the joint angles, this distance measure suffices.

Let $\boldsymbol{q}_{\text{ref}}(\psi^*)$ be the closest point to $\boldsymbol{q}(t)$. Finding ψ^* is defined as the non-linear least squares problem of Equation 6

$$\psi^*(\boldsymbol{q}(t)) = \underset{\Psi_l \le \psi \le \Psi_u}{\mathbf{arg\,min}} \quad D(\psi, \boldsymbol{q}(t)) \tag{6}$$

with cost functions $D(\psi, \boldsymbol{q}(t))$ as in Equation 7.

$$D(\psi, \boldsymbol{q}(t)) = (\boldsymbol{q}(t) - \boldsymbol{q}_{\text{ref}}(\psi))^T (\boldsymbol{q}(t) - \boldsymbol{q}_{\text{ref}}(\psi)). \tag{7}$$

To solve this problem the phase estimation is done using Newtons' algorithm [24], with at least one iteration j each time step t_s. The phase ψ resets to the beginning of the cycle when it reaches the end and vice versa, to remain in $[\Psi_l, \Psi_u]$. The update rule is given by Equation 8.

$$\tilde{\psi}^-_{j+1} = \tilde{\psi}^+_j - \Delta\tilde{\psi}^+_j = \tilde{\psi}^+_j - \frac{D'(\tilde{\psi}^+_j)}{D''(\tilde{\psi}^+_j)}, \tag{8}$$

where $D'(\tilde{\psi}^+_j)$ and $D''(\tilde{\psi}^+_j)$ are the first and second derivative of D to ψ evaluated at $\tilde{\psi}^+_j$. Omitting the dependency on ψ and t:

$$D' = 2\boldsymbol{q}'^T_{\text{ref}}[\boldsymbol{q}'_{\text{ref}} - \boldsymbol{q}]$$

and

$$D'' = 2\boldsymbol{q}'^T_{\text{ref}}\boldsymbol{q}'_{\text{ref}} + 2\boldsymbol{q}''^T_{\text{ref}}[\boldsymbol{q}_{\text{ref}} - \boldsymbol{q}]. \tag{9}$$

Here $\boldsymbol{q}''_{\text{ref}}$ is derived from $\boldsymbol{q}'_{\text{ref}}$ using Equation 28 from appendix A. This algorithm finds the extrema. It is thus also possible that it converges to a maximum when $D'' < 0$. The Gauss-Newton algorithm approximates $D'' \approx 2\boldsymbol{q}'^{T}_{\text{ref}}\boldsymbol{q}'_{\text{ref}} \geq 0$. However, this approximation results in a less accurate estimate of the minimum, causing the estimation to jump between time instances. This creates non-smoothness in the control signal. The used algorithm therefore switches between Newton and Gauss-Newton, displayed in Equation 10.

$$D'' := \begin{cases} D''_N & \text{if } D''_N \geq 0 \\ D''_{GN} & \text{if } D''_N < 0 \end{cases}. \tag{10}$$

When the initial estimate is closest to a local maximum, the algorithm will use Gauss-Newton to converge to the nearest local minimum and converge to this minimum with Newton, when that becomes the nearest extremum.

To go to a next or previous cycle, the following reset is implemented after each iteration:

$$\tilde{\psi}^{+}_{j} := \begin{cases} \tilde{\psi}^{-}_{j} - \Psi_u + \Psi_l & \text{if } \tilde{\psi}^{-}_{j} > \Psi_u \\ \tilde{\psi}^{-}_{j} - \Psi_l + \Psi_u & \text{if } \tilde{\psi}^{-}_{j} < \Psi_l \\ \tilde{\psi}^{-}_{j} & \text{otherwise.} \end{cases} \tag{11}$$

The fact that the algorithm does not find a global minimum is not a concern, as the controller will also attract $\boldsymbol{q}$ to $\boldsymbol{q}'_{\text{ref}}(\psi^*)$. Convergence of the controlled system to the reference therefore guarantees that the estimation becomes the global minimum after a certain amount of time steps.

B. Phase velocity estimation

The speed estimates that are saved during each measurement are approximated using a finite differences method. It disregards the resets, to prevent inaccurate jumps at the domain borders:

$$\dot{\tilde{\psi}}_{j+1} := \frac{\Delta\tilde{\psi}^{+}_{j}}{t_k - t_{k-1}} + \dot{\tilde{\psi}}_{j}. \tag{12}$$

Note that the time increment is calculated as the difference between measurement instances. In contrast to the phase estimate, which updates every time step t_s, the speed estimate is saved during each measurement instance k and set to zero such as in Equation 13.

$$\dot{\tilde{\psi}}_{j} = 0 \quad \forall\, t_j = t_k. \tag{13}$$

The speed could also be estimated as a finite difference over each time step t_s instead of each measurement. However, the above methods allows the estimate to be averaged over the amount of iterations between measurements.

V. Interaction Control

This section presents how the interaction control torque is defined and how the ILC torques are computed

A. Feedback control

The quantization of the joint position encoders leads to inaccurate joint velocity measurements (see Figure 6). The combination with series elastic actuation does not allow smooth velocity reference tracking.

However, damping is possible on the motors directly. The torque trajectory $\boldsymbol{u}_{\text{ref}}(\psi)$ plus the PD-feedback controller give the control torque as in Equation 14.

$$\boldsymbol{u}(\tilde{\psi}, t_s) = \boldsymbol{u}_{\text{ref}}(\tilde{\psi}) + \mathbf{K_p}(\boldsymbol{q}_{\text{ref}}(\tilde{\psi}) - \boldsymbol{q}(t_s)) - \mathbf{K_d}\boldsymbol{\zeta}(t_s), \tag{14}$$

where $\boldsymbol{q}_{\text{ref}}(\tilde{\psi})$ indicates the reference position as in Equation 1 with the phase estimate $\tilde{\psi}$, $\boldsymbol{\zeta}$ indicates the motor angular velocities and $\mathbf{K}_p$ and $\mathbf{K}_d$ indicate hand-tuned diagonal proportional and derivative gain matrices respectively. The torque trajectory $\boldsymbol{u}_{\text{ref}}(\tilde{\psi})$ is trained using the ILC approach as described below.

B. Iterative learning control

A proportional-derivative (PD)-ILC approach [25] trains the feedforward control trajectory $\boldsymbol{u}_{\text{ref}}(\tilde{\psi})$.

The torque vector $\boldsymbol{u}_{ILC,k}(\tilde{\psi}_k)$ is the PD-ILC torque update, which is computed as in Equation 15.

$$\boldsymbol{u}_{ILC,k} = \gamma \boldsymbol{u}_{\text{ref}}(\tilde{\psi}_k) + g_p \boldsymbol{e}_k + g_d \dot{\boldsymbol{e}}_k. \tag{15}$$

Here γ is referred to as the forgetting factor, g_p and g_d are the gains and $\boldsymbol{e}(\tilde{\psi}_k, t_k)$ and $\dot{\boldsymbol{e}}(\tilde{\psi}_k, t_k)$ are the position and velocity errors:

$$\begin{aligned} \boldsymbol{e}_k &= \boldsymbol{q}_{\text{ref}}(\tilde{\psi}_k) - \boldsymbol{q}(t_k) \\ \dot{\boldsymbol{e}}_k &= \dot{\boldsymbol{q}}_{ref}(\tilde{\psi}_k) - \dot{\tilde{\boldsymbol{q}}}(t_k) \end{aligned} \tag{16}$$

with $\dot{\boldsymbol{q}}_{\text{ref}}(\tilde{\psi}_k)$ as in Equation 2 and velocity estimate $\dot{\tilde{\boldsymbol{q}}}(t)$ at time step t_s as in Equation 17.

$$\dot{\tilde{\boldsymbol{q}}}(t_s) = \frac{\boldsymbol{q}(t_s) - \boldsymbol{q}(t_{s-1})}{t_s - t_{s-1}}. \tag{17}$$

To create independence of the ILC algorithm from the damping factor β, g_d and g_p in Equation 15 are set as a function of β, given in Equation 18.

$$\begin{aligned} g_p &= \gamma_p \beta \\ g_d &= \gamma_d \beta. \end{aligned} \tag{18}$$

An elaboration as to why this renders the ILC algorithm approximately independent with respect to β is provided in Appendix B. The forgetting factor is set to $\gamma = 1$ during for the rest of the paper. Further discussion on the forgetting factor is provided in the discussion.

VI. Trajectory update

This section discusses how the data is handled during ILC and IKT periods and how it is used to update the trajectories.

A. Data update

At each measurement k the phase estimate $\tilde{\psi}_k$ is used to fill a new row of a matrix $\mathbf{W}_k$ with $\boldsymbol{w}^T(\tilde{\psi}_k)$, such that the each column refers to its respective control point.

In IKT mode, the positions $\boldsymbol{q}(t_k)$ and estimated phase speed $\dot{\tilde{\psi}}_k$ are saved in a new row of the data matrix $\mathbf{M}_{\boldsymbol{q}\dot{\psi},k}$. In ILC mode a data matrix $\mathbf{M}_{u,k}$ is filled with torque updates $\boldsymbol{u}_{ILC,k}$. Only a limited number of past measurements is used, referred to as the memory size N. Using the position-speed data matrix $\mathbf{M}_{\boldsymbol{q},\dot{\psi},k}$ as an example, the updates of $\mathbf{M}_{\boldsymbol{q},\dot{\psi},k}$ and $\mathbf{W}_k$ look like Equation 19 and 20 respectively.

$$\mathbf{M}_{\boldsymbol{q}\dot{\psi},k} = \begin{bmatrix} \boldsymbol{q}^T(t_{k-N}) & \dot{\tilde{\psi}}_{k-N} \\ \boldsymbol{q}^T(t_{k-N+1}) & \dot{\tilde{\psi}}_{k-N+1} \\ \vdots & \vdots \\ \boldsymbol{q}^T(t_k) & \dot{\tilde{\psi}}_k \end{bmatrix} \tag{19}$$

and

$$\mathbf{W}_k = \begin{bmatrix} \boldsymbol{w}^T(\psi_{k-N}) \\ \boldsymbol{w}^T(\psi_{k-N+1}) \\ \vdots \\ \boldsymbol{w}^T(\psi_k) \end{bmatrix}. \tag{20}$$

Note that the value of N is significantly smaller than the value N_{init} used during initialization, as it is not required for the former that the data covers the complete motion trajectory. A larger N increases the amount of information available each update, such that there are less trials needed to completely update the trajectory, but more computational power is needed every update.

B. Trajectory fit

A damped least squares approach updates the trajectory splines given the incoming data points. Consider a data matrix $\mathbf{M}_k$, which can be either $\mathbf{M}_{\boldsymbol{q},\dot{\psi},k}$ or $\mathbf{M}_{\boldsymbol{u},k}$ and control points estimated at the previous measurement $\mathbf{P}_{k-1}$, which can be either

$$\mathbf{P}_{k-1} = \begin{bmatrix} \mathbf{P}_{\boldsymbol{q},k-1} \\ \mathbf{P}_{\dot{\psi},k-1} \end{bmatrix} \quad \text{or} \quad \mathbf{P}_{k-1} = \mathbf{P}_{\boldsymbol{u},k-1}$$

depending on whether the system is in IKT or ILC mode respectively. Consider also the matrix $\mathbf{W}_k$ from Equation 20. Because we are looking for a spline in the neighborhood of the previous one the least squares estimate of Equation 4 is transformed equivalently to the Tikhonov regularization [26] as in Equation 21.

$$\begin{bmatrix} \mathbf{P}_k \\ \mathbf{\Lambda}_k \end{bmatrix} = \begin{bmatrix} \mathbf{M}_k^T \mathbf{W}_k + \beta \mathbf{P}_{k-1} \\ \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{W}_k^T \mathbf{W}_k + \beta \mathbf{I} & \mathbf{T}^T \\ \mathbf{T} & \mathbf{0} \end{bmatrix}^{-1}, \tag{21}$$

where $\mathbf{I}$ is an appropriately sized identity matrix. The value for β resembles the damping factor of the learning process. It defines the learning rate of the algorithm and is the value that is varied in the experiment. The use of cyclic updates of the trajectory have also been investigated. A comparison between these methods and a notion about the use of an adapted β can be found in Appendix C-A.

VII. Experiment

Two experiments are conducted where the proposed methods are implemented and tested: An **IKT** experiment is performed to investigate the influence of different values of the damping factor β on the learning process. A walking-like motion is taught to one leg of a humanoid robot using the IKT algorithm as presented in the previous sections. An **ILC-only** experiment is done to test the performance of the ILC. To be able to compare the teaching performances of the human for different values of β, the influence of ILC must remain constant. It is therefore investigated whether the ILC performance is constant for values of β.

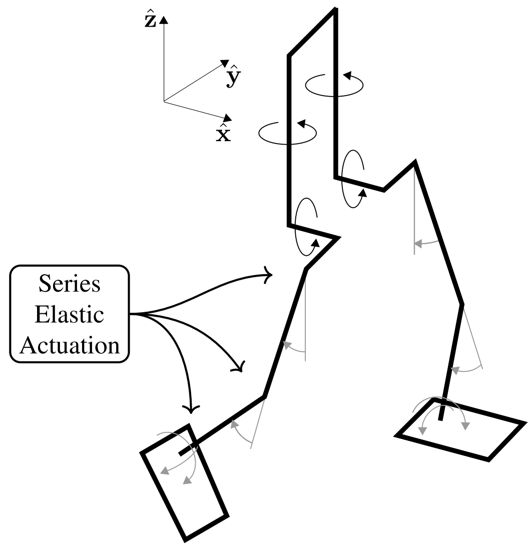


Fig. 2: Diagram of the joint configuration of the TUlip humanoid robot, with the series elastic actuated joints. The zero angles are displayed for the hip and knee joints, which are used in the experiment.

A. Setup

The TUlip humanoid robot, developed at the Delft University of Technology [27], is used to conduct the experiments. It consists of a torso and two legs, weighs $18\,\mathrm{kg}$ and measures $1.1\,\mathrm{m}$. Six joints are in the legs that measure $0.6\,\mathrm{m}$. The joint configuration of the robot is displayed in Figure 2. There are two ankle joints along the transverse and sagittal axes, one knee joint and three hip joints. The knee and ankle joints and hip transversal joint are controlled using series elastic actuation [14]. Relative joint position encoders measure the joint angles. The torque signal is updated at $500\,\mathrm{Hz}$, such that the increment between successive time steps t_s as presented in Section VI is $(t_s - t_{s-1} = 2\,\mathrm{ms})$.

B. Experimental protocol

During the IKT experiment, the author performed the teaching task as is displayed in Figure 3. The trajectory that the robot is supposed to follow after teaching is referred to as the target trajectory. The taught trajectory that the robot aims to

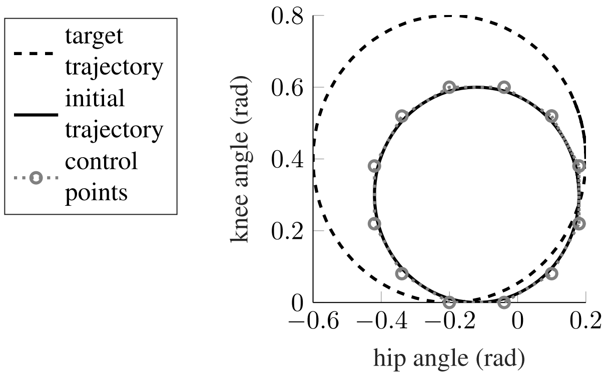


Fig. 3: The initial and target knee-hip trajectories of the right robot leg. The human needs to convert the initial trajectory to the target trajectory: a circular motion with a radius of $0.4\,\mathrm{rad}$. The cycle time is 2 seconds, such that the velocity trajectory is circular with a radius of $0.4\pi\,\mathrm{rad}$. The trajectories are encoded as a 3rd order spline with 12 control points.

follow, is referred to as the reference trajectory. This trajectory is updated based on the demonstrations by the human during IKT. Six test trials and one practice trial are executed for each value of β. To diminish effects of different initializations, each trial starts with the same initial trajectory. After two ILC cycles, the subject can start the demonstration mode.

During the ILC-only experiment the reference trajectory is kept at the initial reference trajectory of Figure 3. One trial is performed for each value of β.

In both experiments, the performance is measured for five values of β : 100, 300, 1000, 3000, 10000 (unitless). The memory size is set at $N = 10$. A measurement is taken every 0.01 seconds ($t_k - t_{k-1} = 0.01\,\mathrm{s}$). The ILC gains are configured as: $g_p = 0.05\beta(\mathrm{N\,m/rad})$ and $g_d = 0.005\beta(\mathrm{N\,m\,s/rad})$. The entries, each corresponding to a joint, on the diagonal of the controller gain matrices $\mathbf{K}_p$ and $\mathbf{K}_d$ are given in Table I.

TABLE I: Feedback gains per joint

Joint	k_p (N m/rad)	k_d (N m s/rad)
Hip Z	10	5
Hip X	75	37.5
Hip Y	45	22.5
Knee Y	25	12.5
Ankle Y	60	15
Ankle X	50	25

C. Data analysis

The average Euclidean distance per cycle between the target and reference trajectory in the joint space for the joint positions and velocities are used to investigate the performance. For the IKT experiment they are indicated as $Q_{\boldsymbol{q}}$ and $Q_{\dot{\boldsymbol{q}}}$ for the position and velocity trajectory respectively. They are defined as in Equation 22, with position and velocity errors $\boldsymbol{\epsilon}_k$ and $\dot{\boldsymbol{\epsilon}}_k$ as in Equation 23.

$$\begin{aligned} Q_{\boldsymbol{q}} &= \frac{\sum_{k=k_{\mathrm{init},l}}^{k_{\mathrm{end},l}} \sqrt{\boldsymbol{\epsilon}_k^T \boldsymbol{\epsilon}_k}}{k_{\mathrm{end},l} - k_{\mathrm{init},l}} \\ Q_{\dot{\boldsymbol{q}}} &= \frac{\sum_{k=k_{\mathrm{init},l}}^{k_{\mathrm{end},l}} \sqrt{\dot{\boldsymbol{\epsilon}}_k^T \dot{\boldsymbol{\epsilon}}_k}}{k_{\mathrm{end},l} - k_{\mathrm{init},l}}. \end{aligned} \tag{22}$$

The variables $k_{\mathrm{init},l}$ and $k_{\mathrm{end},l}$ are the first and last measurement instances of cycle l. The position and velocity errors at measurement instance k are defined as:

$$\begin{aligned} \boldsymbol{\epsilon}_k &= \boldsymbol{q}_{\mathrm{tg}}(\psi_{\mathrm{tg},k}) - \boldsymbol{q}_{\mathrm{ref}}(\tilde{\psi}_k) \\ \dot{\boldsymbol{\epsilon}}_k &= \dot{\boldsymbol{q}}_{\mathrm{tg}}(\psi_{\mathrm{tg},k}) - \dot{\boldsymbol{q}}_{\mathrm{ref}}(\tilde{\psi}_k). \end{aligned} \tag{23}$$

Here $\boldsymbol{q}_{\mathrm{tg}}(\psi_{\mathrm{tg},k})$, $\dot{\boldsymbol{q}}_{\mathrm{tg}}(\psi_{\mathrm{tg},k})$ and $\psi_{\mathrm{tg},k}$ are target position and velocity and the phase estimate along the target trajectory $\boldsymbol{q}_{\mathrm{tg}}(\psi_{\mathrm{tg},k})$ at measurement instance k. The vectors $\boldsymbol{q}_{\mathrm{ref}}(\tilde{\psi}_k)$ and $\dot{\boldsymbol{q}}_{\mathrm{ref}}(\tilde{\psi}_k)$ are the references as in Section III and $\tilde{\psi}_k$ is the phase estimate along the reference trajectory at measurement k. A new cycle begins when $\psi_{\mathrm{tg},k}$ resets. Decreasing values of $Q_{\boldsymbol{q}}$ and $Q_{\dot{\boldsymbol{q}}}$ indicate improvement.

To interpret the values of $Q_{\boldsymbol{q}}$ and $Q_{\dot{\boldsymbol{q}}}$ a metric is introduced as the ratio between $Q_{\boldsymbol{q}}$ and $Q_{\dot{\boldsymbol{q}}}$ and the radius of the position and velocity target trajectories respectively:

$$\begin{aligned} \eta_{\boldsymbol{q}} &= \frac{Q_{\boldsymbol{q}}}{r_{\boldsymbol{q},\mathrm{tg}}} \\ \eta_{\dot{\boldsymbol{q}}} &= \frac{Q_{\dot{\boldsymbol{q}}}}{r_{\dot{\boldsymbol{q}},\mathrm{tg}}}. \end{aligned} \tag{24}$$

For the ILC-only experiment, the performance is measured similarly as in Equation 22 as the average Euclidean distance between the reference states and the actual states, i.e. using the measured errors $\boldsymbol{e}_k$ and $\dot{\boldsymbol{e}}_k$ of Equation 16.

$$\begin{aligned} Q_{\boldsymbol{q},ILC} &= \frac{\sum_{k=k_{\mathrm{init},l}}^{k_{\mathrm{end},l}} \sqrt{\boldsymbol{e}_k^T \boldsymbol{e}_k}}{k_{\mathrm{end},l} - k_{\mathrm{init},l}} \\ Q_{\dot{\boldsymbol{q}},ILC} &= \frac{\sum_{k=k_{\mathrm{init},l}}^{k_{\mathrm{end},l}} \sqrt{\dot{\boldsymbol{e}}_k^T \dot{\boldsymbol{e}}_k}}{k_{\mathrm{end},l} - k_{\mathrm{init},l}}. \end{aligned} \tag{25}$$

VIII. Results

Results of the ILC-only experiment and the IKT experiment are shown in this section.

A. Iterative learning control

The performance metrics for the ILC are displayed in Figure 4. For each damping factor β the variable $Q_{\boldsymbol{q}}$ converges from an error of approximately $0.05\,\mathrm{rad}$ to $0.01\,\mathrm{rad}$ over 26 cycles. The value of $Q_{\dot{\boldsymbol{q}}}$ converges from $0.6\,\mathrm{rad\,s^{-1}}$ to about $0.2\,\mathrm{rad\,s^{-1}}$.

For reference, the trial of the ILC experiment for $\beta = 1000$ is shown in Figures 5 and 6 for the position and velocity respectively. The knee does not extend fully as is shown by the flattened part in the trajectory at the bottom of Figure 5. Figure 6 shows the velocity measurements smoothed with a moving average filter with a window of 3 data points. The

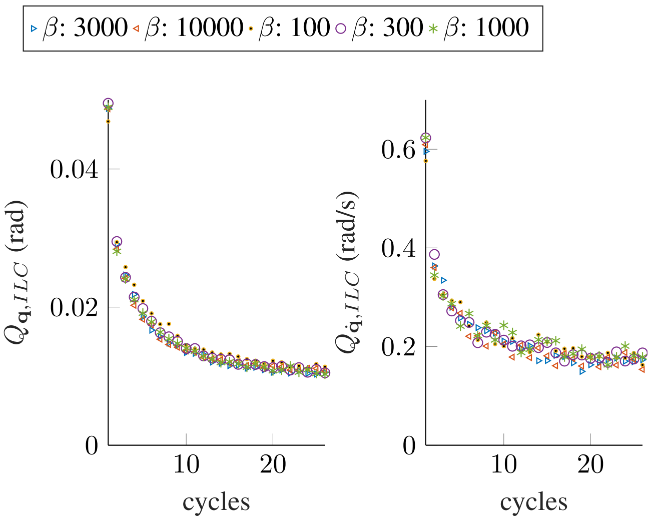


Fig. 4: Performance of the ILC controller for different damping factors β. The reference trajectory is constant over the cycles. The mean per cycle of the root of the sum of squared errors at each measurement are displayed for the position and velocities to the left and the right respectively.

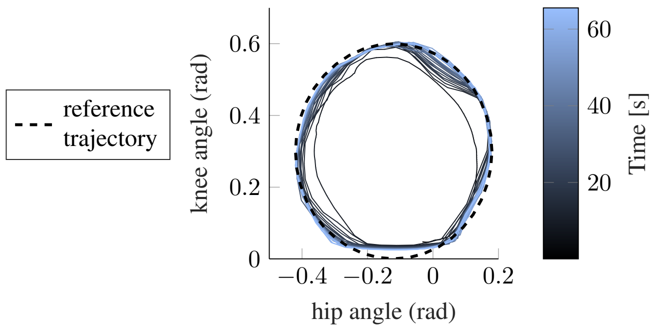


Fig. 5: Position tracking progression of the ILC trial with $\beta = 1000$. The motion direction is counter-clockwise.

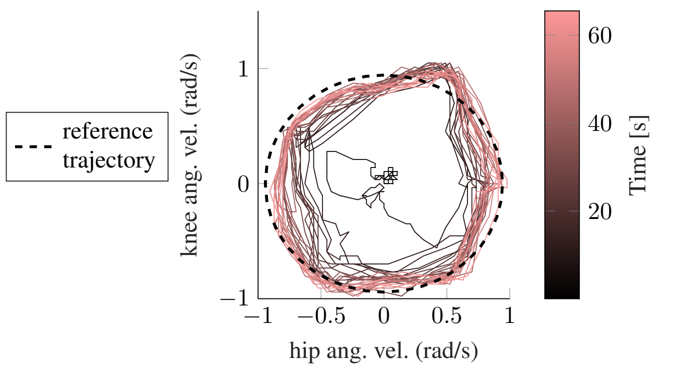


Fig. 6: Smoothed velocity tracking progression of the ILC trial with $\beta = 1000$. The direction of the motion is counter-clockwise.

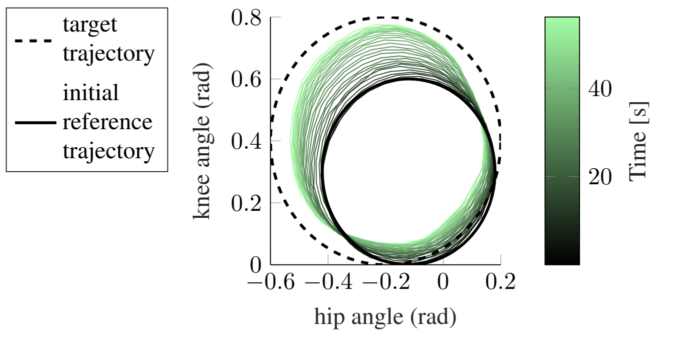


Fig. 7: Position trajectory progression of an exemplary trial, for $\beta = 1000$. The motion direction is counter-clockwise. The final trajectory has a performance of $Q_{\boldsymbol{q}} = 0.055\,\mathrm{rad}$

straight segments of the trajectory are due to the quantization on the velocity measurements. The Figure shows overshoot of the velocity in the top right corner of the trajectory.

B. *Incremental kinesthetic teaching*

Figure 7 shows an exemplary IKT trial. The subject is able to move the center of the cyclic trajectory to coincide with the target trajectory in contrast to the radius. Knee extension decreases over each cycle.

The values of $Q_{\boldsymbol{q}}$ and $Q_{\dot{\boldsymbol{q}}}$ per trial are shown in Figure 8. Figure 9 shows the metrics averaged per value of β. Considering that the radius of the position target trajectory is $0.4\,\mathrm{rad}$, the mean $Q_{\boldsymbol{q}} \approx 0.6\,\mathrm{rad}$ after 26 cycles at $\beta = 1000$ indicates an average position error $\eta_{\boldsymbol{q}}$ of $13\,\%$ over this cycle. The mean of $Q_{\dot{\boldsymbol{q}}} \approx 0.28\,\mathrm{rad\,s^{-1}}$ after 26 cycles at $\beta = 1000$ for the circular velocity trajectory with radius $0.4\pi\,\mathrm{rad\,s^{-1}}$ indicates an error $\eta_{\dot{\boldsymbol{q}}}$ of approximately $18\,\%$.

IX. Discussion

This section is divided into three parts. The first two of which concern the ILC performance and the influence of β on the teaching performance. The last part discusses the general teaching performance of the subject using the presented algorithm.

A. *Iterative learning control performance*

Figure 4 shows that the ILC performance using gains as in Equation 18 does not depend on the damping factor β.

That the controller does not achieve knee flexion, lower than a certain angle, largely influences the final value of $Q_{\boldsymbol{q},ILC}$ as shown in Figures 4 and 5. This indicates resistance in the joint past that angle.

The overshoot in the velocity, visible in Figure 6 indicates that the ILC is not able to counteract high velocities as well as too low velocities. When the robot traverses a part of the trajectory faster, there are less trajectory updates for that part, reducing the amount of training for these parts. Moreover, faster movements result in high frequencies in the dynamics. The density of the control points is too low to counteract these dynamics accurately.

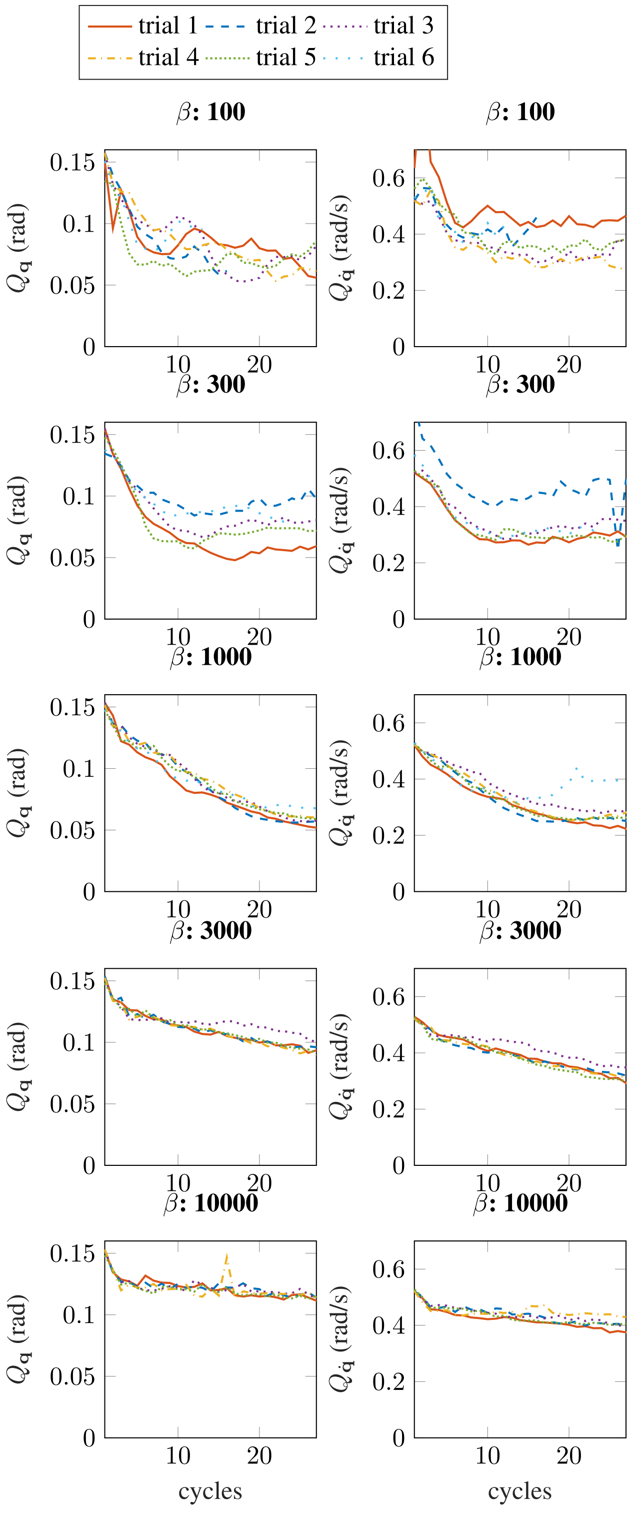


Fig. 8: Values of Q_q and $Q_{\dot{q}}$ for different damping factors β. The memory size is $N = 10$. Six test trials and one practice trial were done per value of β. The test trials are shown. A measurement was done every 0.01 seconds. The shortened trials for $\beta = 100$ indicate learning failures in which extra loops were created in the learned trajectory.

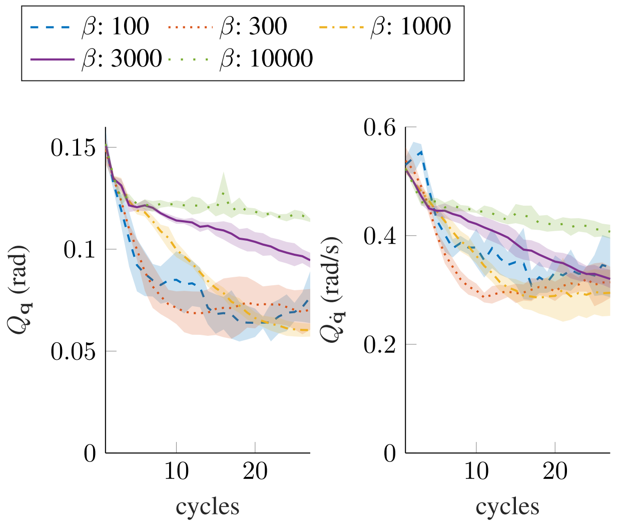


Fig. 9: Values of Q_q and $Q_{\dot{q}}$ for different damping factors β during different trials. The first trial at $\beta = 100$ and second trial at $\beta = 300$ are not included in the mean because of an initialization error. The first trials of $\beta = 1000, 3000, 10000$ are not included to have an equal amount of trials per damping factor. As such, the data represents means of five experiments $\pm 90\,\%$ confidence interval.

B. *The influence of the damping factor on the teaching performance*

As shown in Figure 9 a damping factor of 1000 results in the smallest error between final reference and target trajectory. Lower damping factors allow faster improvements at the beginning. The IKT performance is a trade-off between the final error and the improvement speed.

The experimental task involves combined visual and tactile stimuli. In [28] it was found that the average reaction time of a human on such stimuli is about 0.2 seconds, which corresponds to a tenth of the cycle time of the experiment. Such a reaction is too slow to apply the right force at the right time. This leaves the subject with two ways to accurately position the robot. The first is co-contraction [29], but it is hard to maintain this for many cycles because it requires a lot of energy. The second is by learning a feedforward force trajectory. When the damping factor is low, each new trial influences the behavior of the robot more, such that it reacts increasingly differently each trial and it becomes harder to identify a feedforward force trajectory. The subject is not able to repeat performance between trials, as is indicated by the increased confidence interval for low values of β in Figure 9.

If values for β decreases significantly, it is possible that a mistake by the human deforms the trajectory too much, creating loops in the trajectory. If these loops become too large, it becomes increasingly difficult to smoothen them. This is because more erroneous parts of the trajectory need larger updates, which either requires a larger difference with respect to the new measurement or more measurements. However, the phase estimate is prone to move away from these parts, because it searches for the closest point on the trajectory. The erroneous parts will receive less measurement updates while their neighboring parts will also receive updates that should correct the faulty part. This pulls the neighboring parts together, increasing the respective loop.

C. *The overall teaching performance*

Comparing Figures 7 and 8, it strikes that the learned trajectories under maximal performance are not accurate with respect to the target trajectory.

An important factor for this is that the push button is not intuitive enough in indicating the beginning and end of an IKT period to allow the subject to do targeted improvements, with short IKT periods. This is because the teacher has to hold the button separately, while both hands are needed to position the robot. As such, the process becomes continuous and the subject has to keep track of the complete cycle at all times, which increases the task difficulty. Furthermore, the controller does not update, due to the lack of ILC periods in between the IKT periods.

A limited change of the center of the trajectory does not significantly increase the effort of the teacher. However, an increase in the radius of the cycle does, because it increases the length and velocity of the motion. This explains why the subject corrects the center of the cyclic motion well, in contrary to the radius. ILC periods in between the IKT periods should reduce the control effort from the teacher, such that it becomes easier to increase the radius of the cycle.

Considered that a stable walking motion depends heavily on a few parts of the complete trajectory: the push off and foot placement, this limited performance does not signify that the subject is not suited to teach the robot such a trajectory, as he is allowed to put his focus mainly on these few parts. It points out however, that the task objective should be very limited per IKT period. The subject is only able to teach one leg at the time and can only improve a part of the trajectory each time.

The fact that knee extension decreases during the IKT is partly explained as it is also not achieved by the interaction controller, which indicates the difficulty.

X. Recommendations

The following succeeding research directions are recommended:

A. *Full body limit cycle walking behaviors*

Two recommendations are proposed concerning the learning of full body walking motions.

Initializing the walking motion: A walking cycle involves both legs that can not be taught by a single teacher at the same time. The work of [8] solves this by initializing a full body trajectory from motion capture data. Because the dynamics of the human differ significantly to those of the robot, such a trajectory will not be likely to generate a feasible walking trajectory. A certain quality of the initial walking trajectory is required regarding the balance of the robot, because there are only limited IKT improvements possible each time and they need to be done in series, because different adaptations at the same time create unexpected effects, due to the linked dynamics of the different states. A simulated walking motion might result in an initial trajectory that is closer to a feasible walking motion for the real robot than the motion capture approach. It should be investigated if such a trajectory can be adapted by the human to increase the number of steps the robot is able to make before falling.

Learning different strides: To derive entire walking behaviors from the trajectory, it must be investigated whether new stride motions can be learned from the nominal stride trajectory as this guarantees that the control points of the different trajectories lie in the same phases of the stride. This allows the learned trajectories to be used as motion primitives, which can be combined into a complete walking policy. This then consists of a high level controller in the form of a map between the uncontrolled floating base states and mixing factors that define the combination of the trajectories. As the robot can fall in both horizontal directions and these failure modes are linearly independent, the stabilizing controllers can be decoupled, such that the control problem is divided in two separate one dimensional problems. This results in small state and action spaces, which facilitates effective learning. Furthermore, combinations of the trajectories can be used as starting points for another IKT process, to define extra key points on the map.

It was found that the use of a continuous action space RL does not provide an improvement over discrete ones as the high action frequency averages out the long term response [30], for the much lower stride frequency the use of a continuous value function might improve performance.

B. *Expanding the experimental group:*

The experiment should be repeated for a larger group of test subjects to be able to generalize conclusions on the influence of human teaching performance on the optimal value for β. When the human makes less and smaller mistakes, the trajectory can be updated faster. It is expected that the optimal value of β decreases when the control ability of the human increases due to practice.

C. *The algorithm*

Multiple improvements on the algorithm are suggested:

Free-knot spline fitting: Currently a fixed-knot approach is used estimate the B-spline trajectory. The quality of the method depends heavily on the amount and phase location of the knots, which depends on the form and dimensionality of the trajectory. Free-knot spline fitting methods such as the works of [31], [32] and [33] estimate the amount and location of knots together with their location, such that they do not need to be determined as a design parameter.

Varying the damping factor: To improve the learning speed, it should be investigated if the damping factor can vary during a trial. It should be examined if it is beneficial to begin with a small β to promote fast improvements at the beginning and increase it after a certain amount of trials to guarantee consistent performance at the end.

Varying the update frequency: To increase the accuracy of the estimates at fast or erroneous parts of the trajectory, it is recommended that the measurement and update frequency are a function of the phase speed estimate $\dot{\tilde{\psi}}$ and errors $\boldsymbol{e}$ and $\dot{\boldsymbol{e}}$ of Equations 16. During the ILC periods, this reduces the overshoot in the velocities that is shown in Figure 6 and it would solve the loop-problem as addressed at the end of Subsection IX-B.

Optimizing trajectory updates: Information about objectives of the motion trajectory can be integrated in the spline update in the form of an objective function over which is optimized, or in the form of constraints. An example of this is shown in Appendix D-C. The spline update can be evaluated afterwards such that the spline update is succeeded by an optimization step, or the optimization or constraints can be implemented directly into the damped least squares spline update.

Post-processing of the estimate: To solve the problem of the phase estimation algorithm to disregard the parts of the trajectory that require the most updating, the algorithm might also be extended to evaluate the estimated phases of the measurements, such that weights are spread more evenly over the measurements.

D. The interface

The following recommendations are proposed to improve the interface:

Intuitive mode switching: To make the switching between IKT and ILC more intuitive, it is proposed to indicate the IKT start with a button press and the end with a release, such that the button remains pressed during IKT. Moreover, currently the button needs to be held together with the robot such that such both robot and button need to be handled with one hand. The button should be fixed on the robot next to the joint that is moved by the human teacher, such that it can be reached easily during the IKT. Implementing a robot skin, to detect forces applied by the human would also provide a solution. Furthermore, a more intuitive switching between IKT and ILC enables the human to relax in between IKT periods, such that co-contraction becomes feasible during these periods. Since the human does not have to learn when to apply which forces, teaching should become easier.

Training with multiple teachers: To be able to test with multiple subjects, it is proposed to increase robustness of the hardware.

E. The iterative learning controller

Two recommendations concerning the iterative learning controller are as follows:

Setting the ILC forgetting factor γ: The result of [34] can be used to assess which value to use for γ. As the robot needs to behave predictable for the human to be able to teach it, γ should generally be set higher than the human forgetting factor. However, as there is already damping in the trajectory update, γ could be set somewhat lower. In standard PD-ILC γ is not used though and is therefore set to 1, which is well above the average human forgetting factor of 0.76 as shown in [34].

Investigating ground impacts in ILC: Another recommendation regarding the ILC, is to investigate the influence of varying ground impacts on the ILC torque trajectory and tracking performance. Slowing down the ILC process might improve the damping of varying ground contacts, during stepping. The ILC learning rate is already tuned close to its upper bound, with respect to frequencies of the robot dynamics. As such, speeding up is not feasible.

XI. Conclusion

In the investigation towards robot learning from demonstration for walking robots, this work proposes a model-free incremental kinesthetic teaching method with limited impedance control to teach a series elastic actuated robot a motion trajectory. An iterative learning control method is proposed to learn a phase depends torque trajectory, to allow the velocity of the cycle to be adapted. It is shown that it allows for trajectory tracking when no feedback velocity control is available. The approach was tested on the TUlip humanoid robot using a elliptic walking-like target trajectory.

The iterative learning controller accurately tracked the position trajectory with the exception of a full knee extension. The controller tracked the velocity trajectory better under counteracting forces than under concurring forces.

The subject was able to correct the center of the trajectory but not the radius, because this requires increased effort from the teacher. Due to non-intuitive switching between IKT and ILC tasks, the trials were performed constantly in IKT mode, such that the interaction controller could not increase the reduction of the effort from the teacher. Moreover, it was difficult for the subject to focus on the entire trajectory.

To increase accuracy of the teaching process, a more intuitive interface should be implemented. This allows the human teacher to indicate fast and exactly which parts of the trajectory need demonstrations, for which joints. The human can then focus entirely on the indicated parts. Moreover, an initial walking like trajectory is required such that only small incremental updates are required by the teacher.

Appendix A
B-splines

This section provides background on the B-spline representation and the least-squares fit of a B-spline on a multi-dimensional measurement sequence.

A. B-spline definition

An exemplary B-spline (or basis spline) is shown in Figure 10. A B-spline $\boldsymbol{b}(\psi) \in \mathbb{R}^n$ is an n-dimensional piecewise-polynomial curve that is defined using Equation 26 [21].

$$\boldsymbol{b}(\psi) = \sum_{i=1}^{m} \boldsymbol{p}_i w_{i,o}(\psi) = \mathbf{P}\boldsymbol{w}(\psi), \tag{26}$$

where $\boldsymbol{p}_i \in \mathbb{R}^n$ are m control points that define the shape of the curve and $\psi \in [\Psi_l, \Psi_u)$ is the value that represents a location on the curve (or phase). Ψ_l and Ψ_u are the upper and lower boundaries indicating the beginning and end of the

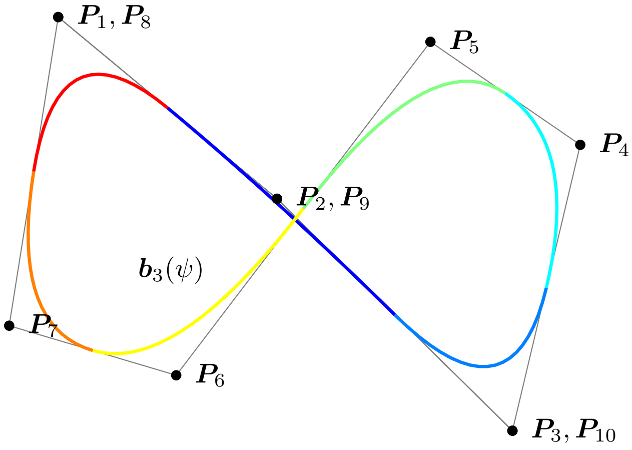


Fig. 10: A 2-dimensional third-order closed B-spline trajectory $\boldsymbol{b}_3(\psi)$ with ten control points $\boldsymbol{P}_i \in \mathbb{R}^2$ with $i \in \{1, \ldots, 10\}$. The phase $\psi \in [0, \ldots, 2\pi)$ indicates the location along the spline. The colors indicate segments of the spline. A third-order indicates that each segment is influenced by three control points and each control point influences three segments. Therefore, the first and last three control points need to overlap to close the loop [35]

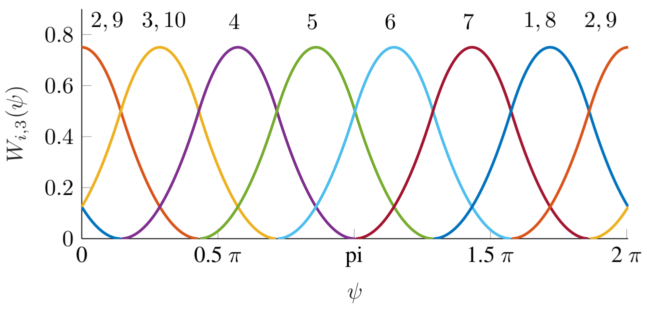


Fig. 11: Third-order B-spline basis functions of a closed spline. They define the influence of each control point $\boldsymbol{p}_i$ on the trajectory at a given location ψ. The integers above the figure indicate the control point instance i.

trajectory. The value o represents the order or degree of the spline, such that an order $o = 1$ indicates linear segments, $o = 2$ indicates quadratic segments and so on. The basis functions $W_{i,o}(\psi) \in \mathbb{R}^m$ are defined using the Cox-de Boor recursion formula [16]. They indicate the influence of the control points on each location of the curve. The basis functions of an exemplary third-order B-spline are provided in Figure 11. The equation in matrix form is given by $\mathbf{P} \in \mathbb{R}^{n\times m} = [\boldsymbol{p}_1, \ldots, \boldsymbol{p}_m]$ and the vector with basis function looks like Equation 27.

$$\boldsymbol{w}(\psi) = \begin{bmatrix} w_1(\psi) \\ \vdots \\ w_m(\psi) \end{bmatrix}. \tag{27}$$

The derivative $\boldsymbol{b}'(\psi)$ of the spline $\boldsymbol{b}(\psi)$ to the phase ψ is computed as Equation 28.

$$\boldsymbol{b}'(\psi) = \sum_{i=1}^{m} \frac{\psi}{\psi_{i+o+1} - \psi_{i+1}} \left(\boldsymbol{p}_{i+1} - \boldsymbol{p}_i\right) W_{i,o} \tag{28}$$

which renders a trajectory of degree $o - 1$ [36].

B. *Fixed-knot multi-dimensional B-spline fitting*

Given the data point matrix $\mathbf{M} \in \mathbb{R}^{N\times n}$, control points matrix $\mathbf{P} \in \mathbb{R}^{n\times m}$ and basis matrix $\mathbf{W} \in \mathbb{R}^{N\times m}$, where N is the amount of data measurements. The entries of $\mathbf{W}$ are computed based on the phase location $\boldsymbol{\psi}_k$ of each measurement $k \in \{1, \ldots, N\}$, using the recursion formula of [16]. To fit the control points $\boldsymbol{P}_j \in \mathbb{R}^{1\times m}$ per dimension $j \in \{1, \ldots, n\}$ on the data points in $\boldsymbol{M}_j \in \mathbb{R}^{N\times 1}$, the least squares cost functions that are minimized for each dimension are [17]:

$$\begin{aligned} J_1 &= (\boldsymbol{M}_1 - \mathbf{W}\boldsymbol{P}_1^T)^T(\boldsymbol{M}_1^T - \mathbf{W}\boldsymbol{P}_1^T) \\ &\vdots \\ J_{n-1} &= (\boldsymbol{M}_{n-1} - \mathbf{W}\boldsymbol{P}_{n-1}^T)^T(\boldsymbol{M}_{n-1} - \mathbf{W}\boldsymbol{P}_{n-1}^T) \\ J_n &= (\boldsymbol{M}_n - \mathbf{W}\boldsymbol{P}_n^T)^T(\boldsymbol{M}_n - \mathbf{W}\boldsymbol{P}_n^T). \end{aligned} \tag{29}$$

Using the least squares, the solutions to these minimizations that follow from setting $\frac{\mathrm{d}\,J_j}{\mathrm{d}\,\boldsymbol{P}_j} = 0$, are given by Equation 30

$$\begin{aligned} \boldsymbol{P}_1 &= \boldsymbol{M}_1^T\mathbf{W}\left(\mathbf{W}^T\mathbf{W}\right)^{-1} \\ &\vdots \\ \boldsymbol{P}_{n-1} &= \boldsymbol{M}_{n-1}^T\mathbf{W}\left(\mathbf{W}^T\mathbf{W}\right)^{-1} \\ \boldsymbol{P}_n &= \boldsymbol{M}_n^T\mathbf{W}\left(\mathbf{W}^T\mathbf{W}\right)^{-1}. \end{aligned} \tag{30}$$

Concatenating these equations, the minimization for all dimensions can be done at once:

$$\begin{aligned} \begin{bmatrix} \boldsymbol{P}_1 \\ \vdots \\ \boldsymbol{P}_{n-1} \\ \boldsymbol{P}_n \end{bmatrix} &= \begin{bmatrix} \boldsymbol{M}_1^T \\ \vdots \\ \boldsymbol{M}_{n-1}^T \\ \boldsymbol{M}_n^T \end{bmatrix} \boldsymbol{W}\left(\mathbf{W}^T\mathbf{W}\right)^{-1} = \\ \mathbf{P} &= \mathbf{M}^T\mathbf{W}\left(\mathbf{W}^T\mathbf{W}\right)^{-1}. \end{aligned} \tag{31}$$

Appendix B
Independence of the Iterative Learning Controller with Respect to the Damping Factor

Consider Equation 21, while disregarding the constraints. The ILC trajectory update then looks like Equation 32.

$$\mathbf{P}_{\boldsymbol{u},k} = \left(\mathbf{M}_{\boldsymbol{u},k}^T\mathbf{W}_k + \beta\mathbf{P}_{\boldsymbol{u},k-1}\right)\left(\mathbf{W}_k^T\mathbf{W}_k + \beta\mathbf{I}\right)^{-1} \tag{32}$$

with position ILC torques from Equation 15 and by ignoring the velocity terms, the matrix $\mathbf{M}_k^T$ for the ILC becomes:

$$\mathbf{M}_{\boldsymbol{u},k}^T = \left[\boldsymbol{u}_{\text{ref}}(\tilde{\psi}_{k-N}), \ldots, \boldsymbol{u}_{\text{ref}}(\tilde{\psi}_k)\right] + g_p\left[\boldsymbol{e}_{k-N}, \ldots, \boldsymbol{e}_k\right]. \tag{33}$$

Equation 18 can now be filled in for g_p. Considering the torque trajectory from Equation 1 and $\mathbf{W}$ from Equation 20, Equation 33 is equal to Equation 34.

$$\mathbf{M}_{\boldsymbol{u},k}^T = \mathbf{P}_{\boldsymbol{u},k-1}\mathbf{W}_k^T + \beta\gamma_p\mathbf{E}, \tag{34}$$

where $\mathbf{E} = [\boldsymbol{e}_{k-N}, \dots, \boldsymbol{e}_k]$. This can be substituted in Equation 32 to form:

$$\begin{aligned}\mathbf{P}_{\boldsymbol{u},k} = &\left(\mathbf{P}_{\boldsymbol{u},k-1}(\mathbf{W}_k^T\mathbf{W}_k + \beta\mathbf{I}) + \beta\gamma_p\mathbf{E}\mathbf{W}_k\right)\cdot \\ &\left(\mathbf{W}_k^T\mathbf{W}_k + \beta\mathbf{I}\right)^{-1} \\ =&\mathbf{P}_{\boldsymbol{u},k-1} + \beta\gamma_p\mathbf{E}\mathbf{W}_k\left(\mathbf{W}_k^T\mathbf{W}_k + \beta\mathbf{I}\right)^{-1}.\end{aligned} \tag{35}$$

Considering that the entries $\mathbf{W}_k$ are basis function with values between one and zero (see appendix A) and the damping factor β is chosen between 100 and 10000 in the experiments, the updated torque trajectory is approximately as in Equation 36.

$$\begin{aligned}\mathbf{P}_{\boldsymbol{u},k} &\approx \mathbf{P}_{\boldsymbol{u},k-1} + \beta\gamma_p\mathbf{E}\mathbf{W}_k\left(\beta\mathbf{I}\right)^{-1} \\ &= \mathbf{P}_{\boldsymbol{u},k-1} + \gamma_p\mathbf{E}\mathbf{W}_k,\end{aligned} \tag{36}$$

such that the update is independent of β if $\beta\mathbf{I} \gg \mathbf{W}_k^T\mathbf{W}_k$.

APPENDIX C
CYCLIC TRAJECTORY UPDATES

This section address cyclic updates of the motion trajectory.

A. *Damped updates*

Next to the damped least squares method trajectory estimation method of Section VI a cyclic update method was investigated. The two methods are referred to as the α- and β-method corresponding to their respective learning rates. This α method updates the full trajectory every cycle, as a linear combination of the current trajectory estimate and previous trajectory. The damping factor α is used as in Equation 37.

$$\mathbf{P}_{l+1} = (1-\alpha)\tilde{\mathbf{P}}_{l+1} + \alpha\mathbf{P}_l, \tag{37}$$

where $\tilde{\mathbf{P}}_{l+1}$ is the spline estimate using Equation 4 and $\mathbf{P}_l$ is the estimate after the previous cycle l. The update rule then becomes:

$$\Delta\mathbf{P}_{l+1} = (1-\alpha)[\tilde{\mathbf{P}}_{l+1} - \mathbf{P}_l]. \tag{38}$$

B. *Comparing cycle and measurement-wise updates*

A disadvantage with respect to the β-method is that the amount of computations needed is not spread out over the measurements, such that they need to be chopped up artificially, which complicates the code. Moreover, the update can be assessed directly during the demonstration in the β-method, as it also influences the future reference signals due to the continuity of the estimated reference trajectory.

To investigate how the measurement-wise updates relate to the cyclic updates in terms of damping, the memory size N of Section VI is increased, such that there are measurements available around the whole trajectory and the weight matrix $\mathbf{W}$ is full column rank. The measurements of the α- and β-method are set equal, as well as the update frequency. If the constraints are disregarded for the sake of the argument, the damping factors of the two methods can be compared when the updates are set equal, such that $\Delta\mathbf{P}_\alpha = \Delta\mathbf{P}_\beta$. This results in Equation 39

$$\boldsymbol{\Gamma}(\mathbf{W}^T\mathbf{W} + \beta\mathbf{I})^{-1} = \boldsymbol{\Gamma}(1-\alpha)(\mathbf{W}^T\mathbf{W})^{-1} \tag{39}$$

with

$$\boldsymbol{\Gamma} = \mathbf{M}^T\mathbf{W} - \mathbf{P}_{k-1}\mathbf{W}^T\mathbf{W}. \tag{40}$$

$$\begin{aligned}(\mathbf{W}^T\mathbf{W} + \beta\mathbf{I})&(\mathbf{W}^T\mathbf{W})^{-1} \\ &= \mathbf{I} + \beta(\mathbf{W}^T\mathbf{W})^{-1} \\ &= (\mathbf{W}^T\mathbf{W})^{-1}(\mathbf{W}^T\mathbf{W} + \beta\mathbf{I})\end{aligned} \tag{41}$$

$$\begin{aligned}\boldsymbol{\Gamma}(\mathbf{W}^T\mathbf{W}) &= \boldsymbol{\Gamma}(1-\alpha)(\mathbf{W}^T\mathbf{W} + \beta\mathbf{I}) \\ &= \boldsymbol{\Gamma}\mathbf{W}^T\mathbf{W} - \boldsymbol{\Gamma}\alpha\mathbf{W}^T\mathbf{W} + \boldsymbol{\Gamma}\beta(1-\alpha)\end{aligned} \tag{42}$$

Equation 41, results in Equation 42, such that:

$$\boldsymbol{\Gamma}\frac{\alpha}{1-\alpha}\mathbf{W}^T\mathbf{W} = \boldsymbol{\Gamma}\beta. \tag{43}$$

This equation does not hold for any $\mathbf{W}$ and $\boldsymbol{\Gamma}$. The above and the fact that the updates of the β-method are not necessarily at each cycle illustrates that the learning rate β can not be related directly to the forgetting factor α.

The previous trajectory is completely overwritten for $\alpha = \beta = 0$ and there is no update for $\alpha = 1$ that corresponds to $\beta \to \infty$.

C. *Forgetting factors*

To assess which values of either β or α are optimal for the average human, we refer to the work of [34]. They investigate human rehabilitation by a robot, which can be seen as the opposite of the approach of this work. They find that the forgetting factor of the robot needs to be less than that of the human, such that the robot updates his action faster than a human to remain challenging. It seems natural that the forgetting factor in the opposite case needs to be higher than that of the human, such that the robot learns slower than the human. The factors α and β in this work refer to changes in the trajectory instead of the force, but supposing that the process is the similar, the two factors do still not have a similar effect. This is because, in the case of [34] the forgetting factor of the robot causes his action to be unpredictable enough to let the human learn from its actions. In our case, setting it high enough would make its actions predictable such that the human knows what it is doing. However, because there is no exploration in the IKT algorithm, actions of the robot do not become unpredictable in case of a small forgetting factor and the robot does not need to be challenged. However, the result of [34] can be used for the iterative learning controller.

The value for β must remain sufficiently high, such that the condition of the weight matrix remains sufficient. It has also been analyzed if β could be updated in a similar fashion as the damping factor the Levenberg-Marquardt algorithm [37]. Here the damping would vary depending on the residuals of Equation 44. However, this is not suitable as adaptation. When the teacher thinks the trajectory is good, he will deviate little from the estimated trajectory and the residuals will be small. Decreasing the damping will increase the update, such that the trajectory will be updated more none the less, interfering with the teachers intentions.

D. *Measurement weights in the spline fit.*

The use of different weights on the measurements has been investigated briefly in simulation, depending on how far they are in the past. The damped least squares problem of Equation 44 would then be transformed into Equation 44 with $\mathbf{\Gamma}$ as in Equation 45 such that $\gamma_{k-i+1} > \gamma_{k-i}$.

$$\begin{bmatrix}\mathbf{P}_k \\ \mathbf{\Lambda}_k\end{bmatrix} = \begin{bmatrix}\mathbf{M}_k^T\mathbf{W}_k + \beta\mathbf{P}_{k-1} \\ \mathbf{0}\end{bmatrix} \begin{bmatrix}\mathbf{W}_k\mathbf{\Gamma}\mathbf{W}_k^T + \beta\mathbf{I} & \mathbf{T}^T \\ \mathbf{T} & \mathbf{0}\end{bmatrix}^{-1} \tag{44}$$

$$\mathbf{\Gamma} = diag([\gamma_{k-1}, \gamma_{k-2}, \dots, \gamma_{k-N}]), \tag{45}$$

The influence was not significant however, with respect to the case as presented in the experiments.

Appendix D
Counteracting persistent errors

This section discusses how the human can be aided in the teaching process. It considers errors induced by external disturbance, which are separated in errors that vary and errors that persist over different cycles. The section presents two approaches to counteract persistent errors in the learning process. The first method corrects the reference trajectory using an error measurement trial. The second introduces the use of prior knowledge concerning the desired behavior of the robot to adjust the trajectory after each update.

A. *Errors in incremental kinesthetic teaching*

Errors that prevent an optimal learning process are divided into those that vary between cycles and those that persist over different cycles.

Fluctuating errors are either due to external disturbances or errors of the human teacher. Damping the trajectory updates counteracts these effects. They are also counteracted by the teacher during subsequent cycles.

Persisting errors arise for example due to gravity. It is difficult to maintain the torso height of the robot over multiple cycles. Other examples include friction, Mass inertia that weakens curves or play in the joints causing oscillations. Erroneous demonstrations decrease support from the controller, making it increasingly difficult to counteract them.

Moreover, PD feedback control does not counteract persisting errors, resulting in a discrepancy between the performed and demonstrated motion. The teacher can account for this by introducing an offset between the demonstrated and the target motion. However, this increases difficulty of the teaching task and is not always possible due to joint limits.

B. *Error sampling trial*

The first method to counteract persisting errors estimates their influence by doing an error measurement cycle between every certain amount of demonstration cycles. Consider the position trajectory control points $\mathbf{P}_{l+1,d}$ and error sample trajectory $\mathbf{P}_{l+1,e}$ and let the previous trajectory be given by $\mathbf{P}_l$. The change in the trajectory due to the persistent errors is estimated as:

$$\Delta\mathbf{P}_{l+1,e} = \mathbf{P}_{l+1,e} - \mathbf{P}_l. \tag{46}$$

The change in the trajectory during a demonstration is estimated as in Equation 47.

$$\Delta\mathbf{P}_{l+1,d} = \mathbf{P}_{l+1,d} - \mathbf{P}_l, \tag{47}$$

such that trajectory update is done as in Equation 48.

$$\mathbf{P}_{l+1} = \alpha\left(\Delta\mathbf{P}_{l+1,d} - \Delta\mathbf{P}_{l+1,e}\right) + \mathbf{P}_l, \tag{48}$$

where χ is the update weight as in Appendix C.

However, the method does not filter out the persistent errors accurately, because the teacher corrects implicitly for some of them by showing the target motion. The robot is not certain whether a demonstrated motion is to counteract the errors or because it is the target motion. The method over-corrects.

C. *Assistance using performance information*

The second method uses knowledge in terms of correct behavior, to asses what kind of correction is needed. The correct behavior is encoded as an optimization function, that depends on the control points $\mathbf{P}_{\boldsymbol{q}}$ of joint trajectory. A gradient descent-like method reduces the function after every trajectory update.

Consider as example the torso height z. When z decreases every demonstration inverse kinematics can be used to increase the mean of z over the trajectory.

$$z_{mean} = f(\mathbf{P}_{\boldsymbol{q}}, \mathbf{P}_{fb}) \tag{49}$$

such that:

$$\frac{\delta z_{mean}}{\delta \boldsymbol{P}_{\boldsymbol{q}}} = \bar{\mathbf{J}}_z^T(\mathbf{P}_{\boldsymbol{q}}, \mathbf{P}_{fb}) \tag{50}$$

where $\bar{\bullet}$ indicates the vectorization of a matrix, $\mathbf{J}_z^T(\mathbf{P}_{\boldsymbol{q}}, \mathbf{P}_{fb})$ is the jacobian of z_{mean} with respect to the joint configuration control points $\mathbf{P}_{\boldsymbol{q}}$ as a function of the control points of the estimated floating base state trajectory $\mathbf{P}_{fb}$. Equation 50 reduces the height error $e_z = z_{mean} - z_{ref}$:

$$\mathbf{P}_{\boldsymbol{q}}^+ = \mathbf{P}_{\boldsymbol{q}}^- - e_z\eta_z\bar{\mathbf{J}}_z^T(\mathbf{P}_{\boldsymbol{q}}^-, \mathbf{P}_{fb}) \tag{51}$$

This example is not model free, as a kinematic model of the robot is needed to estimate the height of the torso. The floating base coordinate points $\mathbf{P}_{fb}$along the trajectory are estimated using information of the robot's joint positions, ground contact switches and IMU data, using the extended Kalman filter, as shown in [38] for example.

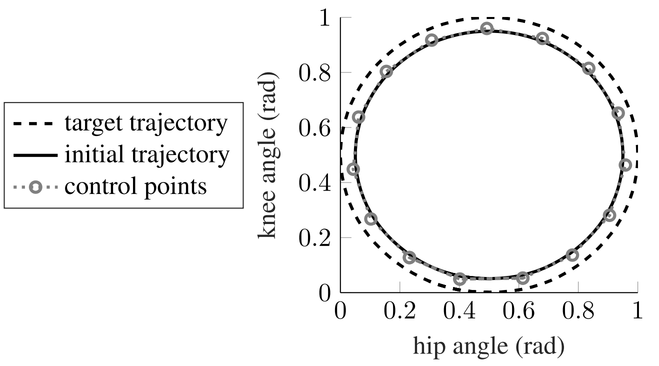


Fig. 12: The initial and target knee-hip trajectories for the simulated experiment. The human needs to convert the initial trajectory to the target trajectory. The trajectories are encoded as a 3rd order spline with 15 control points. Zero angles indicate that the hip is straight down and the knee is completely extended. Positive angles indicate knee flexion and hip extension.

Appendix E
Incremental kinesthetic teaching on a simulated robot

The experiment as presented in Section VII is also done in simulation using a fixed base model of one leg of the TUlip robot. The influence of different values for the memory size N and the damping factor β are investigated.

A. Experimental setup

The human needs to adapt an initial hip-knee trajectory as shown in Figure 12. Each trial starts with one cycle in the ILC learning mode. After this the human can press the first button of the joystick to start the IKT mode and press it again to return to the ILC mode. A `Logitech Force 3D Pro`™ joystick is used to disturb the leg. Pushing the joystick to the front and back (negative and positive along joystick axis 1) indicated positive and negative knee torques respectively. Left and right joystick pushes (negative and positive along joystick axis 2) indicate negative and positive hip torques respectively.

B. Discussion

The results are shown in Figure 13. The optimum of β is due to a trade off between learning rate and stability. For values of β higher than its optimum, the learning rate decreases, but the learning process remains stable. This means that the performance measures will converge to the same level for higher values of β, given enough trials. For values of β smaller than the optimum, the learning process becomes less stable. For low damping factors, mistakes by the human are not damped out. This indicates that the value optimal value of β decreases with the control ability of the human. The physical interaction between the human and the robot is likely to increase this ability, such that lower optimal values and thus faster convergence are expected in the experiment.

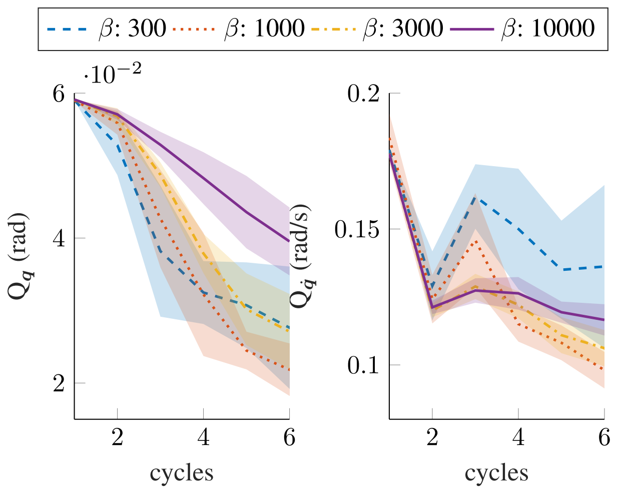


Fig. 13: Values of Q_{pos} and Q_{vel} for different damping factors β. It was considered to use the beta values of $\beta = 100, 300, 1000, 3000, 10000$, but at $\beta = 100$ the trajectory got mangled, such that the subject was not able to complete the trial. The knee and hip input torques are computed as $u_{hip} = 2\tau_2$ and $u_{knee} = -2\tau_1$, where τ_1 and τ_2 are the joystick angles of axes 1 and 2 respectively. The a memory size is $N = 10$. A measurement was done every 0.05 seconds. The velocity trajectory does not improve for $\beta = 300$. The fastest convergence is at $\beta = 1000$. The data represents means of five experiments $\pm$ 95% confidence interval.

Appendix F
Piecewise-affine trajectory following with vector fields

This section discusses an alternative piecewise trajectory encoding method and how it is used to create state-dependent reference tracking.

A. Encoding the motion as a piecewise affine trajectory

The idea is based on [39]. In this work a vector field is created using a piecewise approximation of the trajectory with a finite number n of points $\boldsymbol{p}_i$, connected by affine segments $\boldsymbol{l}$, such that $\boldsymbol{l}_i$ connects points $\boldsymbol{p}_i$ and $\boldsymbol{p}_{i+1}$ The vector field consists of subfield $\boldsymbol{V}_i$ corresponding to each segment $\boldsymbol{l}_i$. Each subfields is an affine system consisting of a contraction part perpendicular to the segment as in Equation 52 to force the robot to the trajectory and part that flows parallel to the segment.

$$\begin{aligned} \boldsymbol{V}_i &= \boldsymbol{V}_{\perp i} + \boldsymbol{V}_{\parallel i} \qquad \text{with} \\ \boldsymbol{V}_{\perp i} &\Rightarrow \dot{\boldsymbol{x}} = \mathbf{A}_{\perp i}\boldsymbol{x} + \boldsymbol{b}_{\perp i} \\ \boldsymbol{V}_{\parallel i} &\Rightarrow \dot{\boldsymbol{x}} = \boldsymbol{b}_{\parallel i}, \end{aligned} \tag{52}$$

where $\boldsymbol{x}$ is the state vector and $\mathbf{A}_{\perp i}$ and $\boldsymbol{b}_{\perp i}$ are the affine system matrices of the contraction field and $\boldsymbol{b}_{\parallel i}$ determines the parallel field. To provide continuity, the instantaneous field is a linear combination of two active subfields:

$$\boldsymbol{V} = (1 - \chi)\boldsymbol{V}_i + \chi\boldsymbol{V}_{i+1}. \tag{53}$$

Here $0 \leq \chi \leq 1$ is zero when $\boldsymbol{x}$ lies besides the midpoint of $\boldsymbol{l}_i$ and it goes to one when $\boldsymbol{x}$ is next to the midpoint of $\boldsymbol{l}_{i+1}$, when this happens the active segments switch $(i = i + 1)$. In [39] affinity of the vector field is used to calculate chi in advance by solving the equations of motion for the vector field. The reference is state-dependent. To create a state-dependent reference χ must depend on $\boldsymbol{x}$. This is done using Equation 54.

$$\begin{aligned} \chi &= \frac{\Delta_{i+2} - \Delta_{i+1}}{\Delta_{i+2} - \Delta_i} \quad \text{with} \\ \Delta_j &= \|\boldsymbol{p}_j - \boldsymbol{x}\|. \end{aligned} \tag{54}$$

The switching is realized by setting $\Delta_{i+2} < \Delta_i \Rightarrow i = i + 1$. When a force pushes the robot back along the trajectory, a switch back is also possible such that $\Delta_{i-1} < \Delta_{i+1} \Rightarrow i = i - 1$.

The use of this piecewise affine approximation has the disadvantage that the reference tracking fails when angles between the piecewise affine segments become sharp er than 90°, as is shown in [39].

B. State dependent tracking of the piecewise trajectory

The dynamics resulting from the use of the algorithm of the previous subsection is defined by the vector field parameters and controller parameters. To decrease the amount of tuning parameters the use of position control provides contraction towards a segment. It is transformed to work perpendicular to the trajectory. Velocity reference tracking creates a force in the direction of the trajectory. Setting the closest point of the trajectory to current configuration of the robot as reference, removes the need of a piecewise affine approximation.

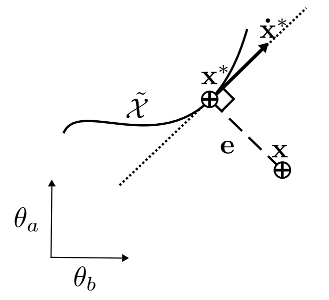


Fig. 14: The figure shows the joint configuration $\boldsymbol{x}$ and linear trajectory segment $\mathrm{l}_{i,j}$. The desired velocity vector $\dot{\tilde{\mathbf{x}}}$ at $\tilde{\boldsymbol{x}}$, lies along tangent $\mathcal{T}\tilde{\boldsymbol{X}}$ on $\tilde{\boldsymbol{x}}$. $\boldsymbol{e}$ is thus perpendicular to $\dot{\boldsymbol{x}}^*$.

Let a trajectory be defined as the manifold $\tilde{\mathcal{X}} \subset \mathbb{S} = \{\tilde{\boldsymbol{x}} | \tilde{\boldsymbol{x}} = \boldsymbol{b}_{\boldsymbol{q}}(\psi) | \psi \in [\Psi_l, \Psi_u]\}$ with $\boldsymbol{x}$ as the current joint configuration of the robot (see Figure 14). Position-wise it is satisfactory if $\boldsymbol{x} \in \tilde{\mathcal{X}}$. The controller achieves this by minimizing position error that is the distance between the joint configuration and its closest point on the trajectory in terms of the Euclidean distance: $\boldsymbol{e} = \boldsymbol{q}_{\text{ref}}(\psi^*) - \boldsymbol{x}$.

As shown in Figure 14 the component of $\boldsymbol{e}$ in the direction of $\dot{\boldsymbol{x}}^*$ is always zero, such that the component of actuating forces and torques due to the position tracking of $\boldsymbol{x}^*$ never act in the direction of the velocity reference tracking of $\dot{\boldsymbol{x}}^*$. No transformation of the position control is required.

Looking at the above, it might be noticed that the weighted activation of two consecutive segments of the piecewise affine trajectory is actually a way of approximating $\boldsymbol{q}_{\text{ref}}(\psi^*)$:

$$\boldsymbol{q}_{\text{ref}}(\psi^*) \approx (1 - \chi)\boldsymbol{p}_j + \chi \boldsymbol{p}_{j+1}. \tag{55}$$

Appendix G
A model-based stepping trajectory

The following section provides notes on the model-based design of a walking motion.

A. Stepping phases

Consider a bipedal robot, consisting of a trunk and two legs with six floating body and twelve joint degrees of freedom (DOF). Each footstep of the robot is separated into the following four phases depending on the ground contact conditions.

1) **Double stance** Where both feet are kept on the ground, constraining $6 \times 2 = 12$ DOF.
2) **Push off** The back foot rotates around its front edge, such that the amount of constrained DOF becomes $6 \times 2 - 1 = 11$
3) **Swing** The back foot is free and moves to a new position in front of the other foot, such that 6 DOF remain constrained.
4) **Heel Strike** The swing foot heel is in contact with the ground, while the toes are still in the air

After heel strike the role of the feet switch and the robot can either go back to double stance, or go to the push off phase directly, when its center of mass (COM) has enough forward momentum. At each phase the remaining DOF need to be controlled to follow some trajectory or go to some point, in such a way that the ground contact constrains are not violated. As the robot would fall down in the latter case.

The presented method is to design a rough trajectory for the the six DOF (3 rotational and 3 linear) of the trunk and one for every unconstrained DOF of the swing foot.

B. Linearized hybrid dynamics

A suboptimal one-step-ahead MPC controller is designed to convert the desired trunk and foot trajectories to feasible joint trajectories, by solving a quadratic program (QP). Different methods can be found in the literature (e.g. [40]) that linearize the robotic system and design an LQR approach to stabilize the system. However, these linearizations require derivatives of the gravitational and Coriolis matrices $\mathbf{V}(\boldsymbol{q}(k))$ and $\mathbf{C}(\boldsymbol{q}(k), \dot{\boldsymbol{q}}(k))$ with respect to the discrete states $\boldsymbol{q}(k)$ and $\dot{\boldsymbol{q}}(k)$ at discrete time instance k to be defined. While computing the derivative of $\mathbf{V}(\boldsymbol{q}(k))$ is doable but computationally extensive for systems with a large dimensionality, computing the derivative of $\mathbf{C}(\boldsymbol{q}(k), \dot{\boldsymbol{q}}(k))$ is remains cumbersome. Operating points are mostly set to have zero velocity to cancel out the the Coriolis forces altogether. In case of a walking motion, this assumption is not accurate. Using a one-step-ahead MPC control approach does not require derivatives of the matrices of the EOM of the robotic system as defined in Equation 56 with respect to the states, because they are approximately constant during one time step. Moreover, the discretization allows rigid contact dynamics to be incorporated in the dynamic system.

Consider the equations of motion (EOM) of the free floating robot as Equation 56.

$$\boldsymbol{M}(\boldsymbol{q}(k))\ddot{\boldsymbol{q}}(k) = \mathbf{C}(\boldsymbol{q}(k), \dot{\boldsymbol{q}}(k))\dot{\boldsymbol{q}}(k) + \boldsymbol{V}(\boldsymbol{q}(k)) + \mathbf{B}\boldsymbol{u}(k) \tag{56}$$

The state is defined by the positions of the DOF q and their velocities $\dot{\boldsymbol{q}}$. The reduced mass matrix is given by $\mathbf{M}(\boldsymbol{q})$, the Coriolis terms by $\mathbf{C}(\boldsymbol{q}, \dot{\boldsymbol{q}})$, gravitational pull $\mathbf{V}(\boldsymbol{q})$ and input matrix $\mathbf{B}$ with joint torques $\boldsymbol{u}$. Forward Euler is used to discretize these equation as it creates a linear dependency on the unknown velocity prediction $\dot{\boldsymbol{q}}(k+1)$:

$$\begin{aligned}\mathbf{M}(\boldsymbol{q}(k))\dot{\boldsymbol{q}}(k+1) = \mathbf{M}(\boldsymbol{q}(k))\dot{\boldsymbol{q}}(k) + h\mathbf{C}(\boldsymbol{q}(k), \dot{\boldsymbol{q}}(k))\dot{\boldsymbol{q}}(k) \\ + h\mathbf{V}(\boldsymbol{q}(k)) + h\mathbf{B}\boldsymbol{u}(k)\end{aligned} \tag{57}$$

where the time $t = hk$ for step k and step size h. For a small time step, the matrices that depend on the positions and velocities are assumed to be constant.

Rigid ground contact constraints are added depending on the step phase. As the stance foot is not allowed to leave the ground, vertical velocity constraints are added using Lagrange multipliers $\boldsymbol{\lambda}_v(k+1)$. $\boldsymbol{\lambda}_v(k+1)$ needs to be positive as it represents the ground forces. Moreover, constraints are added using a linear approximation of the friction cone with the horizontal ground impulses being given by $\boldsymbol{\lambda}_h(k+1)$ and the friction coefficient, given by μ to prevent scuffing. This results in the equalities of Equation 58.

$$\begin{aligned}&\mathbf{M}(k)\dot{\boldsymbol{q}}(k+1) + \mathbf{J}(k)^T\boldsymbol{\lambda}_v(k+1) \\ &\quad +\mathbf{J}_h(k)^T\boldsymbol{\lambda}_h(k+1) - h\mathbf{B}\boldsymbol{u}(k) = \mathbf{M}(k)\dot{\boldsymbol{q}}(k) \\ &\qquad\qquad + h\mathbf{C}(k)\dot{\boldsymbol{q}}(k) + h\mathbf{V}(k) \\ &\mathbf{J}_v(k)\dot{\boldsymbol{q}}(k+1) = \mathbf{0} \\ &\mathbf{J}_h(k)\dot{\boldsymbol{q}}(k+1) = \mathbf{0} \\ &\boldsymbol{\lambda}_v \leq \mathbf{0} \\ &-\mu\boldsymbol{\lambda}_v(k+1) - \boldsymbol{\lambda}_h(k+1) \leq \mathbf{0} \\ &-\mu\boldsymbol{\lambda}_v(k+1) + \boldsymbol{\lambda}_h(k+1) \leq \mathbf{0}.\end{aligned} \tag{58}$$

Here, $\mathbf{J}_v(k)$ and $\mathbf{J}_h(k)$ are respectively the Jacobians relating the velocities perpendicular and parallel to ground to the velocity states.

C. Computing joint torques and trajectories from trunk and swing foot positions

The Euclidean space robot trunk and swing foot positions $\boldsymbol{s}$ velocities $\dot{\boldsymbol{s}}$ need to follow some space trajectory with respect to the stance foot. Towards this goal, minimal controller torques $\boldsymbol{u}$ need to be found that get the trunk and swing foot as close as possible to the desired trajectory.

The position is discretized using backward Euler:

$$\dot{\boldsymbol{s}}(k+1) = \frac{\boldsymbol{s}(k+1) - \boldsymbol{s}(k)}{h}. \tag{59}$$

Because $\boldsymbol{s}(k+1)$ needs to be as close as possible to the desired position $\boldsymbol{s}_d$, the desired velocity towards the trajectory is set to Equation 60.

$$\dot{\boldsymbol{s}}_{d\perp}(k) = \frac{\boldsymbol{s}_d(k) - \boldsymbol{s}(k)}{h}. \tag{60}$$

For a suitably defined reference the trajectory and position references are always perpendicular, as is stated in Appendix F-B. The desired velocity along the trajectory is defined as $\dot{\boldsymbol{s}}_{d,||}(k)$, such that the desired velocity is given by $\dot{\boldsymbol{s}}_d(k) = \dot{\boldsymbol{s}}_{d,\perp}(k) + \dot{\boldsymbol{s}}_{d,||}(k)$. The value to be minimized is given by Equation 61.

$$\begin{aligned}\underset{\boldsymbol{u}}{\mathbf{arg\,min}} \quad & (\dot{s}(k+1) - \dot{s}_d(k))^T\mathbf{Q}_{\boldsymbol{s}}(\dot{s}(k+1) - \dot{s}_d(k)) \\ & + \boldsymbol{u}^T(k)\mathbf{R}\boldsymbol{u}(k)\end{aligned} \tag{61}$$

with $\mathbf{Q}_{\boldsymbol{s}}$ and $\mathbf{R}$ as tuning parameters. The unknown vector $\boldsymbol{x}$ is concatenated as in Equation 62.

$$\boldsymbol{x} = \begin{bmatrix}\dot{\boldsymbol{q}}(k+1) \\ \boldsymbol{u}(k) \\ \boldsymbol{\lambda}_v(k+1) \\ \boldsymbol{\lambda}_h(k+1)\end{bmatrix}. \tag{62}$$

This results in the following QP-problem:

$$\begin{aligned}\boldsymbol{x}^* = \underset{\boldsymbol{x}}{\mathbf{arg\,min}}\ & \boldsymbol{x}^T\mathbf{T}^T(k)\mathbf{Q}\mathbf{T}(k)\boldsymbol{x} - \boldsymbol{c}^T(k)\mathbf{Q}\mathbf{T}(k)\boldsymbol{x} \\ \mathbf{s.t.}\quad & \mathbf{C}(k)\mathbf{x} = \boldsymbol{d}(k) \\ & \mathbf{A}(k)\mathbf{x} \leq \boldsymbol{b}(k)\end{aligned} \tag{63}$$

with

$$\begin{aligned}\mathbf{Q} &= \begin{bmatrix}\mathbf{Q}_{\boldsymbol{s}} & \mathbf{0} \\ \mathbf{0} & \mathbf{R}\end{bmatrix} \\ \mathbf{T}(k) &= \begin{bmatrix}\mathbf{J}_{\boldsymbol{s}}(k) & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{I} & \mathbf{0} & \mathbf{0}\end{bmatrix} \\ \boldsymbol{c}(k) &= \begin{bmatrix}\dot{\boldsymbol{s}}_{d,\perp}(k) + \dot{\boldsymbol{s}}_{d,||}(k) \\ \mathbf{I} \\ \mathbf{0} \\ \mathbf{0}\end{bmatrix} \\ \mathbf{C}(k) &= \begin{bmatrix}\mathbf{M}(k) & -h\mathbf{B} & \mathbf{J}_v(k)^T & \mathbf{J}_h(k)^T \\ \mathbf{J}_v(k) & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{J}_h(k) & \mathbf{0} & \mathbf{0} & \mathbf{0}\end{bmatrix} \\ \boldsymbol{d}(k) &= \begin{bmatrix}\mathbf{M}(k)\dot{\boldsymbol{q}}(k) + h\mathbf{C}(k)\dot{\boldsymbol{q}}(k) + h\mathbf{V}(k) \\ \mathbf{0} \\ \mathbf{0}\end{bmatrix} \\ \mathbf{A} &= \begin{bmatrix}\mathbf{0} & \mathbf{0} & -\mu\mathbf{I} & -\mathbf{I} \\ \mathbf{0} & \mathbf{0} & -\mu\mathbf{I} & +\mathbf{I} \\ \mathbf{0} & \mathbf{0} & -\mathbf{I} & \mathbf{0}\end{bmatrix} \\ \boldsymbol{b} &= \begin{bmatrix}\mathbf{0} \\ \mathbf{0} \\ \mathbf{0}\end{bmatrix}.\end{aligned}$$

It can be solved by `MATLAB`'s *quadprog*, however [40] proposes an easily implementable active-set algorithm that is about 10 times faster than using *quadprog*. Moreover such an algorithm can be compiled such that it can be implemented on a robot in the form of an `xPC`-target.

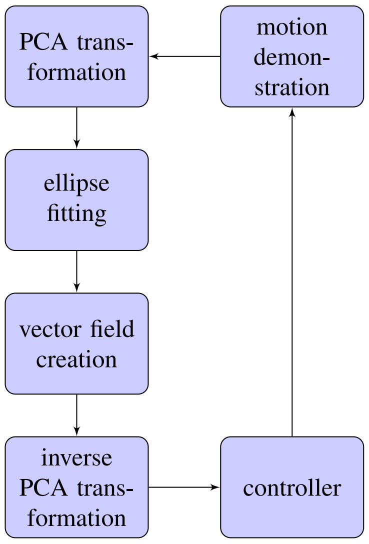


Fig. 15: Scheme of IKT scheme with an elliptical motion representation. The motion's first two principle components are estimated and an ellipse is fitted on their scores. A vector field is created to converge to the elliptical limit cycle. It determines the control reference. The reference is transformed back to work space using the inverse PCA transformation.

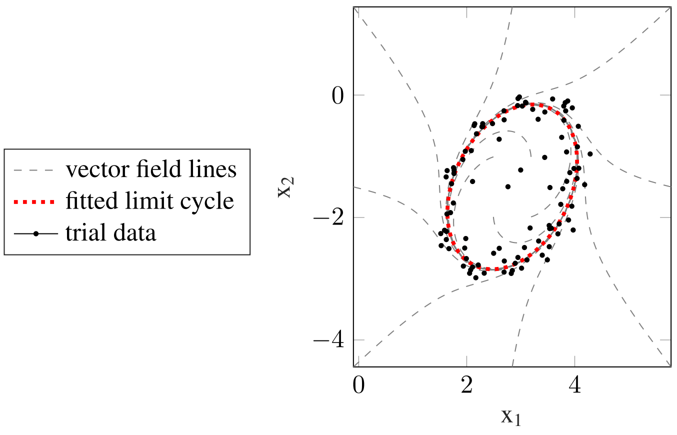


Fig. 16: Fit of an elliptical vector field on trial data. An ellipse is fitted using the method as presented by [41]. The vector field is created using a generalization of the work of [42] that creates a circular vector field, used in robot soccer to circumvent obstacles.

Appendix H
Eliptical vector field

An alternative approach to encode a cyclic reference trajectory as an elliptic vector field is presented in this section. The method is displayed in Figures 15 and 16.

The ellipse fitting method outputs the elliptical best fit on given data as parameters of the implicit Equation 64.

$$ax_1^2 + bx_1x_2 + cx_2^2 + dx_1 + ex_2 + f = 0 \tag{64}$$

with variables x_1 and x_2 and parameters a, b, c, d, e and f [43]. As the ellipse is a transformation of the unit circle, these parameters can be converted to a rotation, translation and scaling of the description of the circle as displayed in Equation 65

$$\begin{aligned}
\alpha &= \begin{cases} \arctan\left(\frac{c-a-\zeta}{b}\right) & \text{for } b \neq 0 \\ 0 & \text{for } b = 0,\ a < c \\ \pi/2 & \text{for } b = 0,\ a > c \end{cases} \\
\boldsymbol{\beta} &= \begin{bmatrix} \frac{2cd-be}{D} \\ \frac{2ae-bd}{D} \end{bmatrix} \\
\boldsymbol{\gamma} &= \begin{bmatrix} -\frac{\sqrt{2\xi(a+c+\zeta)}}{D} \\ -\frac{\sqrt{2\xi(a+c-\zeta)}}{D} \end{bmatrix}
\end{aligned} \tag{65}$$

with $\zeta = \sqrt{(a-c)^2 + b^2}$, $\xi = ae^2 + cd^2 - bde + Df$ and $D = b^2 - 4ac$ [43]. Here α is the angle of rotation, and $\boldsymbol{\beta}$ and $\boldsymbol{\gamma}$are the translation and scaling vectors for x_1 and x_2 direction.

The equations of motion that govern this vector field of [42] are described in affine form as in Equation 66.

$$\begin{aligned}
\dot{\bar{\boldsymbol{y}}} &= \sigma\bar{\mathbf{A}}(\boldsymbol{y})\bar{\boldsymbol{y}} \\
&= \begin{bmatrix} \omega(2-\bar{\boldsymbol{y}}^T\bar{\boldsymbol{y}}) & 1 & 0 \\ -1 & \omega(2-\bar{\boldsymbol{y}}^T\bar{\boldsymbol{y}}) & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ 1 \end{bmatrix},
\end{aligned} \tag{66}$$

where σ determines the velocity and ω determines how quick the system converges to the limit cycle. The elliptical vector field is computed as a similarity transformation [44] with states $\bar{\boldsymbol{x}} = \mathbf{T}\bar{\boldsymbol{y}}$, with homogeneous matrix:

$$\mathbf{T} = \left[\begin{array}{c|c} \text{diag}(\boldsymbol{\gamma}) \begin{bmatrix} \cos\alpha & -\sin\alpha \\ \sin\alpha & \cos\alpha \end{bmatrix} & \boldsymbol{\beta} \\ \hline \mathbf{0} & 1 \end{array}\right].$$

Differentiating $\bar{\boldsymbol{x}}$ gives $\dot{\bar{\boldsymbol{x}}} = \dot{\mathbf{T}}\bar{\boldsymbol{y}} + \mathbf{T}\dot{\bar{\boldsymbol{y}}} = \mathbf{T}\dot{\bar{\boldsymbol{y}}}$. Substituting $\bar{\boldsymbol{y}} = \mathbf{T}^{-1}\bar{\boldsymbol{x}}$ and $\dot{\bar{\boldsymbol{y}}} = \mathbf{T}^{-1}\dot{\bar{\boldsymbol{x}}}$ into Equation 66 gives that the equations of motion are given as in Equation 67.

$$\begin{aligned}
\dot{\bar{\boldsymbol{x}}} &= \mathbf{T}\mathbf{A}(\bar{\boldsymbol{y}})\mathbf{T}^{-1}\bar{\boldsymbol{x}} \quad \text{with} \\
\bar{\boldsymbol{y}} &= \mathbf{T}^{-1}\bar{\boldsymbol{x}}.
\end{aligned} \tag{67}$$

References

[1] J. Peters, S. Vijayakumar, and S. Schaal, "Reinforcement learning for humanoid robotics," in *Proceedings of the third IEEE-RAS international conference on humanoid robots*, pp. 1–20, 2003.

[2] C. G. Atkeson and S. Schaal, "Robot learning from demonstration," in *ICML*, vol. 97, pp. 12–20, 1997.

[3] A. Billard and D. Grollman, "Robot learning by demonstration," *Scholarpedia*, vol. 8, no. 12, p. 3824, 2013. revision #138061.

[4] J. Kober and J. R. Peters, "Policy search for motor primitives in robotics," in *Advances in neural information processing systems*, pp. 849–856, 2009.

[5] P. Kormushev, S. Calinon, and D. G. Caldwell, "Robot motor skill coordination with em-based reinforcement learning," in *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*, pp. 3232–3237, IEEE, 2010.

[6] D. H. Grollman and A. Billard, "Donut as i do: Learning from failed demonstrations," in *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pp. 3804–3809, IEEE, 2011.

[7] S. Calinon and A. G. Billard, "What is the teacher's role in robot programming by demonstration?: Toward benchmarks for improved learning," *Interaction Studies*, vol. 8, no. 3, pp. 441–464, 2007.

[8] D. Kuli, C. Ott, D. Lee, J. Ishikawa, and Y. Nakamura, "Incremental learning of full body motion primitives and their sequencing through human motion observation," pp. 1–16, 2011.

[9] S. Calinon and A. Billard, "Incremental Learning of Gestures by Imitation in a Humanoid Robot," 2007.

[10] S. Calinon and A. Billard, "Active teaching in robot programming by demonstration," in *Robot and Human interactive Communication, 2007. RO-MAN 2007. The 16th IEEE International Symposium on*, pp. 702–707, IEEE, 2007.

[11] D. Lee and C. Ott, "Incremental Motion Primitive Learning by Physical Coaching Using Impedance Control," pp. 4133–4140, 2010.

[12] C. Ott, B. Henze, and D. Lee, "Kinesthetic teaching of humanoid motion based on whole-body compliance control with interaction-aware balancing," pp. 4615–4621, 2013.

[13] S. R. Eddy, "Hidden markov models," *Current opinion in structural biology*, vol. 6, no. 3, pp. 361–365, 1996.

[14] G. A. Pratt and M. M. Williamson, "Series elastic actuators," in *Intelligent Robots and Systems 95.'Human Robot Interaction and Cooperative Robots', Proceedings. 1995 IEEE/RSJ International Conference on*, vol. 1, pp. 399–406, IEEE, 1995.

[15] J. Van der Weijde, "Control design for robust limit-cycle walking," 2014.

[16] C. de Boor, "On Calculating with B-Splines," vol. 62, pp. 50–62, 1972.

[17] P. Dierckx, *Curve and surface fitting with splines*. Oxford University Press, 1995.

[18] K. Levenberg, "A method for the solution of certain non-linear problems in least squares," *Quarterly of applied mathematics*, vol. 2, no. 2, pp. 164–168, 1944.

[19] K. L. Moore, "Iterative learning control: An expository overview," in *Applied and computational control, signals, and circuits*, pp. 151–214, Springer, 1999.

[20] J. Zhang, C. C. Cheah, and S. H. Collins, "Experimental comparison of torque control methods on an ankle exoskeleton during human walking,"

[21] R. H. Bartels, J. C. Beatty, and B. A. Barsky, *An introduction to splines for use in computer graphics and geometric modeling*. Morgan Kaufmann, 1987.

[22] M. Frank and P. Wolfe, "An algorithm for quadratic programming," *Naval Research Logistics (NRL)*, vol. 3, no. 1-2, pp. 95–110, 1956.

[23] B. Siciliano, L. Sciavicco, L. Villani, and G. Oriolo, *Robotics: modelling, planning and control*. Springer Science & Business Media, 2010.

[24] H. Wang, J. Kearney, and K. Atkinson, "Robust and efficient computation of the closest point on a spline curve," in *Proceedings of the 5th International Conference on Curves and Surfaces*, pp. 397–406, 2002.

[25] H.-S. Lee and Z. Bien, "Study on robustness of iterative learning control with non-zero initial error," *International Journal of Control*, vol. 64, no. 3, pp. 345–359, 1996.

[26] A. N. Tikhonov, "On the stability of inverse problems," in *Dokl. Akad. Nauk SSSR*, vol. 39, pp. 195–198, 1943.

[27] T. De Boer, "Foot placement in robotic bipedal locomotion," 2012.

[28] J. V. Hanson, D. Whitaker, and J. Heron, "Preferential processing of tactile events under conditions of divided attention: Effects of divided attention on reaction time," *Neuroreport*, vol. 20, no. 15, p. 1392, 2009.

[29] Mosby, *Mosby's Medical Dictionary*. Elsevier, 9 ed., 2009.

[30] L. Busoniu, R. Babuska, B. De Schutter, and D. Ernst, *Reinforcement learning and dynamic programming using function approximators*, vol. 39. CRC press, 2010.

[31] D. L. Jupp, "Approximation to data by splines with free knots," *SIAM Journal on Numerical Analysis*, vol. 15, no. 2, pp. 328–343, 1978.

[32] M. J. Lindstrom, "Penalized estimation of free-knot splines," *Journal of Computational and Graphical Statistics*, vol. 8, no. 2, pp. 333–352, 1999.

[33] I. DiMatteo, C. R. Genovese, and R. E. Kass, "Bayesian curve-fitting with free-knot splines," *Biometrika*, vol. 88, no. 4, pp. 1055–1071, 2001.

[34] J. L. Emken, R. Benitez, and D. J. Reinkensmeyer, "Human-robot cooperative movement training: learning a novel sensory motor transformation during walking with robotic assistance-as-needed," *Journal of NeuroEngineering and Rehabilitation*, vol. 4, no. 1, p. 8, 2007.

[35] C. K. Shene, "B-spline Curves: Closed Curves." Department of Computer Science, Michigan Technological University, https://pages.mtu.edu/ shene/COURSES/cs3621/NOTES/spline/B-spline/bspline-curve-closed.html.

[36] M. Rogina, *Algebraic Proof of the B-Spline Derivative Formula*, pp. 273–282. Dordrecht: Springer Netherlands, 2005.

[37] D. W. Marquardt, "An algorithm for least-squares estimation of nonlinear parameters," *Journal of the society for Industrial and Applied Mathematics*, vol. 11, no. 2, pp. 431–441, 1963.

[38] X. Xinjilefu, "State estimation for humanoid robots," tech. rep., CARNEGIE-MELLON UNIV PITTSBURGH PA ROBOTICS INST, 2015.

[39] T. Nierhoff, S. Hirche, and Y. Nakamura, "Laplacian trajectory vector fields for robotic movement imitation and adaption," in *Mechanisms and Machine Science*, vol. 22, pp. 205–213, 2014.

[40] S. Kuindersma, F. Permenter, and R. Tedrake, "An efficiently solvable quadratic program for stabilizing dynamic locomotion," in *Robotics and Automation (ICRA), 2014 IEEE International Conference on*, pp. 2589–2594, IEEE, 2014.

[41] A. W. Fitzgibbon, M. Pilu, R. B. Fisher, F. Hill, and E. Eh, "Direct Least Squares Fitting of Ellipses," pp. 1–5.

[42] D.-H. Kim and J.-H. Kim, "A real-time limit-cycle navigation method for fast mobile robots and its application to robot soccer," *Robotics and Autonomous Systems*, vol. 42, no. 1, pp. 17–30, 2003.

[43] W. Besant, *Conic Sections: Treated Geometrically*. Cambridge mathematical series, G. Bell, 1907.

[44] R. Bronson, *Matrix methods: an introduction*. Gulf Professional Publishing, 1991.