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

Solving Motion Tasks with Challenging Dynamics by Combining Kinodynamic Motion Planning and Iterative Learning Control

Michael Meindl[1*], Ferdinand Campe[2*], Dustin Lehmann[2], Thomas Seel[1]

Abstract— **This work considers the problem of robots with challenging dynamics having to solve motion tasks that consist in transitioning from an initial state to a goal state in an environment that is obstructed by obstacles. We propose a novel combination of methods from motion planning and iterative learning control to solve these motion tasks. The proposed method only requires an approximate, linear model of the nonlinear, possibly underactuated robot dynamics. The proposed method employs the approximate, linear model in a kinodynamic rapidly exploring random tree to plan a state trajectory that solves the motion task. Based on the distance to the obstacles, the most relevant samples of the planned trajectory are selected as reference points. Lastly, point-to-point iterative learning control is employed to learn a feedforward input trajectory that leads to the state trajectory precisely tracking the reference points despite the robot's nonlinear real-world dynamics. The proposed method is validated in real-world experiments on a two-wheeled inverted pendulum robot that has to solve a motion task that requires the robot to perform an agile motion to dive beneath an obstacle.**

I. Introduction

Motion tasks are a common occurrence in a variety of application domains, such as production systems [1], mobile robotics [2], traffic systems [3] or biomedical engineering [4]. Motion tasks are characterized by the problem of transitioning a dynamic system from an initial state to a desired target state while maneuvering through an environment. Solving a motion task usually involves two steps: (1) planning the motion to get from the initial to the desired state and (2) executing the planned motion by exciting the inputs of the dynamic system. Often times these two steps can be approached separately and in some applications one of the steps is almost trivial to solve: In the first case, if the environment and the dynamics are simple, planning a motion is trivial (e.g. driving a car in a straight line from point A to B). In the second case, if the dynamics are simple and fully actuated or the planned motion is slow, the execution part is simple and can be done via trivial feed-forward control or by using feedback control.

However, if the environment is complex or the dynamics are constrained, finding a suitable path from the initial state to the desired state is not trivial, e.g. navigating a car through an obstacle course. To solve such a motion planning problem, a variety of approaches have been proposed, which can generally be categorized into two groups: First, there are optimization-based approaches as, e.g., in [5] the Graphs of Convex Sets algorithm was proposed to solve kinodynamic planning problems using convex optimization and applied to a robotic manipulator grasping and placing objects on a shelf. Second, there are sampling-based approaches, as, e.g., in [6] kinodynamic RRT* was proposed and applied to quadrotors navigating an environment occupied by obstacles. These approaches have in common that the planning process usually either considers simple dynamics of the underlying system or requires extensive model knowledge. To overcome the problem, where the real-world dynamics differ from the dynamics used for planning, typically feedback control is used to execute the desired motion [7]. However, feedback control is inherently limited when it comes to performing fast and agile motions [8].

Consequently, if the planned motion is fast or the dynamics are complex, finding the correct input to the underlying dynamic system to execute the desired motion is not straightforward and feedback control might not be suitable. In cases where a desired trajectory is known (either by design or by a motion planning algorithm), a learning-based control algorithm can be used to learn the correct input to the dynamic system to perform the desired motion. One commonly used learning algorithm is Iterative Learning Control (ILC) [9]. ILC has been successfully applied to solve a variety of challenging reference tracking tasks in real-world applications, such as, e.g., controlling a neuroprothesis for a droop foot [4] or quadrotors performing high-speed maneuvers [10]. However, ILC-based methods require a-priori knowledge of the desired trajectory, i.e. not making them suitable stand-alone methods for solving motion tasks. Only a few exceptions such as, e.g., [11] or [12] have extended ILC to be capable of optimizing motions w.r.t., e.g., time or energy efficiency. However, these methods cannot be applied to systems that are affected by sampling-based constraints as is the case for robots that move in environments that are occupied by obstacles.

Motion tasks involving constrained environments and complex dynamics might require the use of both steps, i.e. planning a possible motion and learning the corresponding input on the real system to perform the desired motion. Especially if the system's dynamics are underactuated, i.e. the dynamic system has fewer inputs than outputs, solving any motion problem is a challenge: Planning a motion is difficult, because only a subset of output trajectories is

*Both of these authors contributed equally.

[1]Michael Meindl and Thomas Seel are with the Institute of Mechatronic Systems, Leibniz University of Hannover, 30823 Garbsen, Germany {michael.meindl, thomas.seel}@imes.uni-hannover.de

[2]Ferdinand Campe was and Dustin Lehmann is with the Control Systems Group, Technische Universität Berlin, 10587 Berlin, Germany {f.campe, dustin.lehmann}@tu-berlin.de. Ferdinand Campe is now employed by Technische Universität Braunschweig.

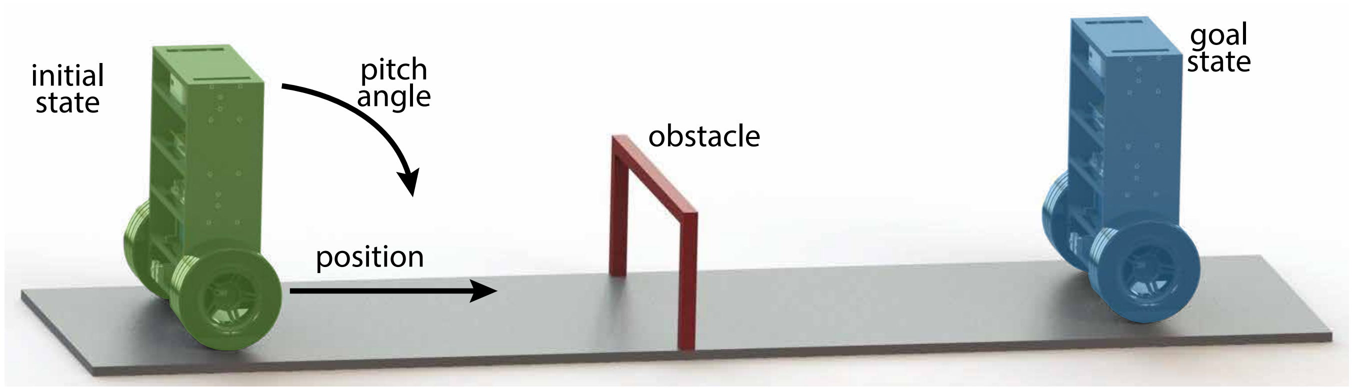

Fig. 1. In this work, we consider robots with challenging dynamics that have to solve motion tasks in environments that are constrained by obstacles. In the example of the two-wheeled inverted pendulum robot (TWIPR), the robot has underactuated, nonlinear dynamics and has to perform a dynamic diving motion in order to transition from the initial state to the goal state without colliding with the obstacle.

feasible, i.e. not every output trajectory can physically be tracked by the dynamic system, so the planning process must already consider the underlying dynamics. Furthermore, executing the planned motion on the real-world system will usually require some adaptation, because planning employs a model of the dynamics which generally deviates from the real-world dynamics.

In this paper, we propose a hybrid method that combines sampling-based motion planning and ILC to solve complex motion tasks in real-world environments. The proposed method employs an approximate, linear model of the underlying (possibly nonlinear, non-minimum phase and/or underactuated) dynamics in a kinodynamic rapidly exploring random tree (RRT) to find a feasible state trajectory. A subset of samples of the planned state trajectory are extracted as reference points and point-to-point iterative learning control is employed to learn an input trajectory that leads to the actual state trajectory tracking the reference points. The method is validated by real-world experiments on a two-wheeled inverted pendulum robot (TWIPR) that has to perform a motion task that requires the robot to dive beneath an obstacle (see Fig. 1). The robot's dynamics are nonlinear, underactuated, and non-minimum phase and the task requires a swift and complex motion to be solved. We show that the proposed method is able to combine both steps of solving motion tasks successfully, making it a suited candidate for a standalone approach for solving motion tasks, even for underactuated systems.

II. Problem Formulation

Consider a dynamic system with state $\mathbf{x} \in \mathbb{R}^n$ and configuration $\mathbf{y} \in \mathbb{R}^m$, which, on each time step $k \in \mathbb{N}$ and trial $j \in \mathbb{N}$, relate to one another according to

$$\mathbf{y}_j(k) = \mathbf{C}\mathbf{x}_j(k)\,. \tag{1}$$

We assume the dynamics to be discrete-time, nonlinear, and repetitive, i.e.,

$$\mathbf{x}_j(k+1) = \boldsymbol{f}(\mathbf{x}_j(k), \mathbf{u}_j(k))\,, \tag{2}$$

where $u \in \mathbb{R}^r$ is the input, and the input and state are constrained to convex sets, i.e., $\mathbf{x} \in \mathcal{X}$ and $\mathbf{u} \in \mathcal{U}$. We assume that the dynamical system corresponds to a robot moving in an either two- or three-dimensional *real-world* which is occupied by obstacles. The robot's position and orientation in the real-world is determined by its configuration $\mathbf{y}$, and the space of admissible configurations, i.e., configurations that do *not* lead to collisions with obstacles, is denoted by $\mathcal{C}$.

To define the motion task, let

$$\bar{\mathbf{u}} := \begin{bmatrix}\mathbf{u}^\top(1) & \mathbf{u}^\top(2) & \dots & \mathbf{u}^\top(N)\end{bmatrix}^\top \tag{3}$$

denote an input trajectory and let

$$\bar{\mathbf{x}} := \begin{bmatrix}\mathbf{x}^\top(1) & \mathbf{x}^\top(2) & \dots & \mathbf{x}^\top(N)\end{bmatrix}^\top \tag{4}$$

denote the state trajectory that results from applying the input trajectory $\bar{\mathbf{u}}$ to the dynamics (2) with $\mathbf{x}(1) = \mathbf{x}_\mathrm{I}$. Now, the motion task consists in finding an input trajectory $\bar{\mathbf{u}}$ that leads to a state trajectory $\bar{\mathbf{x}}$ that ends in the goal state, i.e., $\mathbf{x}(N) = \mathbf{x}_\mathrm{G}$, and all of the corresponding configurations are admissible, i.e., $\forall k,\ \mathbf{y}(k) \in \mathcal{C}$. To learn the desired motion in the real world without obstacle collisions, we assume that the robot also has access to a *practice environment* which is free of obstacles. Once the desired motion has been learned, the result can then be transferred to the environment with obstacles.

We further assume that the dynamics are possibly underactuated, i.e., the dimension of the configuration is larger than the number of inputs, $r \leq n$, and we assume that the nonlinear state dynamics function is not precisely known, but only an approximate linear model of the form

$$\forall j \in \mathbb{N}, k \in \mathbb{N}, \quad \mathbf{x}_j(k+1) = \mathbf{A}\mathbf{x}_j(k) + \mathbf{B}\mathbf{u}_j(k) \tag{5}$$

is known, where $\mathbf{A} \in \mathbb{R}^{n\times n}$ and $\mathbf{B} \in \mathbb{R}^{n\times r}$.

III. Proposed Method

To solve the given problem we propose the following method that consists of three major components, see Figure 2: First, we employ kinodynamic motion planning using the linear model to determine a state trajectory that solves

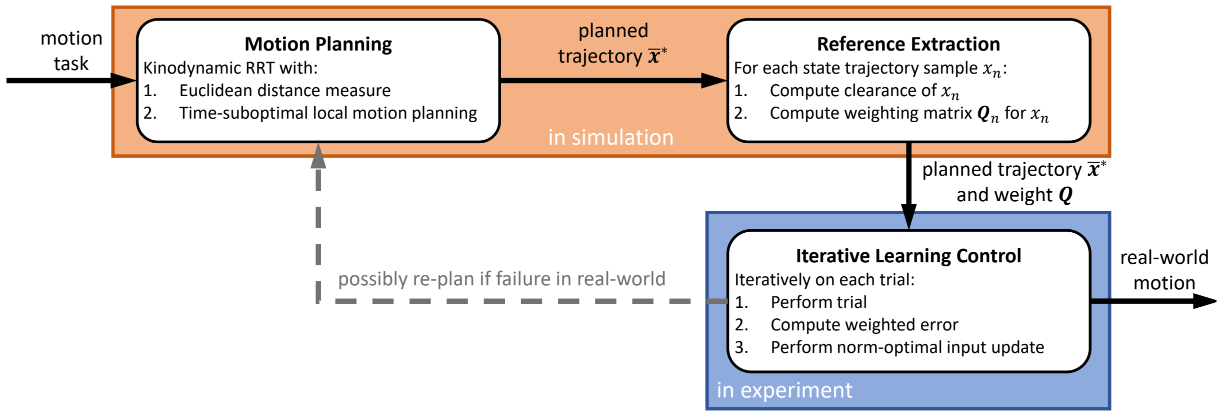
motion task
Motion Planning
Kinodynamic RRT with:
1. Euclidean distance measure
2. Time-suboptimal local motion planning
planned trajectory $\bar{x}^*$
Reference Extraction
For each state trajectory sample x_n:
1. Compute clearance of x_n
2. Compute weighting matrix Q_n for x_n
in simulation
planned trajectory $\bar{x}^*$ and weight Q
Iterative Learning Control
Iteratively on each trial:
1. Perform trial
2. Compute weighted error
3. Perform norm-optimal input update
in experiment
real-world motion
possibly re-plan if failure in real-world

Fig. 2. We propose to combine kinodynamic motion planning and iterative learning control to solve motion tasks despite challenging dynamics. First, a kinodynamic RRT uses an approximate, linear model to find a state trajectory that solves the motion task. Second, the planned trajectories samples are weighted and selected as reference based on their distance to the obstacle. Third, iterative learning control is employed to learn to track the weighted reference trajectory under the real-world dynamics and conditions.

the motion task. Second, the most relevant samples of the planned state trajectory are extracted and used as reference. Third, we employ point-to-point iterative learning control to learn to track the extracted reference points in the real-world environment to solve the motion task.

To solve the motion planning, we employ kinodynamic RRT, whereby the main parameters are the distance function between two states and the local motion planning procedure. In order to measure the distance between two state samples $\mathbf{x}_1,\ \mathbf{x}_2 \in \mathcal{X}$, we employ the Euclidean distance

$$d(\mathbf{x}_1, \mathbf{x}_2) = \sqrt{(\mathbf{x}_1 - \mathbf{x}_2)^\top (\mathbf{x}_1 - \mathbf{x}_2)}\,. \tag{6}$$

For the local motion planning, we employ a procedure to compute a time-suboptimal solution for connecting two states $\mathbf{x}_1$ and $\mathbf{x}_2$ in an efficient manor. For this purpose, assume that at time step k the state equals $\mathbf{x}_1$, then according to the linear dynamics (5), the state at time step $k+n$ is given by

$$\mathbf{x}(k+n) = \mathbf{A}^n \mathbf{x}(k) + \mathbf{G}_n \bar{\mathbf{u}}_n\,, \tag{7}$$

where $\mathbf{G}_n$ is the matrix

$$\mathbf{G}_n = \begin{bmatrix}\mathbf{B} & \mathbf{AB} & \dots & \mathbf{A}^{n-1}\mathbf{B}\end{bmatrix} \tag{8}$$

and $\bar{\mathbf{u}}_n$ is the input trajectory

$$\bar{\mathbf{u}}_n = \begin{bmatrix}\mathbf{u}^\top(n+k) & \mathbf{u}^\top(n+k-1) & \dots & \mathbf{u}^\top(k)\end{bmatrix}^\top. \tag{9}$$

If we want the state on sample $k+n$ to equal $\mathbf{x}_2$, the corresponding input trajectory is computed by

$$\mathbf{u}_n = \mathbf{G}_n^\dagger (\mathbf{x}_2 - \mathbf{A}^n \mathbf{x}_1)\,, \tag{10}$$

where $\mathbf{G}_n^\dagger$ is the pseudo-inverse of $\mathbf{G}_n$. To now compute a time-suboptimal input trajectory that connects $\mathbf{x}_1$ and $\mathbf{x}_2$, we iteratively compute $\mathbf{u}_\mathrm{n}$ for $n \in [1, 2, ...]$ and check, whether the corresponding state and input trajectory violate the constraints. For the first n, in which no constraints are violated, we return $\bar{\mathbf{u}}_n$ as the solution to the local motion planning.

The RRT plans an input trajectory $\bar{\mathbf{u}}^*$ that, if applied to the linear dynamics (5) starting from the initial state $\mathbf{x}_\mathrm{I}$, leads to the planned state trajectory $\bar{\mathbf{x}}^*$ which ends in the goal state $\mathbf{x}_\mathrm{G}$. However, if the planned input trajectory was applied to the real-world dynamics (2), a state trajectory different from $\bar{\mathbf{x}}^*$ would result and, hence, the motion task may not be solved.

To find an input trajectory $\bar{\mathbf{u}}$ that also solves the motion task under the real-world dynamics (2), we first compute the matrix $\mathbf{Q} \in \mathbb{R}^{Nm \times Nm}$ which extracts and weights the most important samples of the planned state trajectory $\bar{\mathbf{x}}^*$ to use them for reference tracking. For this purpose, we first compute the clearance $c(\mathbf{x})$, which is the minimum distance between the robot and the obstacle in real-world coordinates for a given state $\mathbf{x}$, for each sample of the planned state trajectory. Based on the clearance, we differentiate three cases. First, if the clearance of a sample is above the threshold value $\bar{c} \in \mathbb{R}$, the sample of the planned trajectory is not considered in the tracking problem, and it is assigned the weight $\mathbf{0}$, i.e.,

$$c(\mathbf{x}_n) \geq \bar{c} \quad \Longrightarrow \quad \mathbf{Q}_n = \mathbf{0}\,. \tag{11}$$

Second, if the clearance of a sample is below the threshold value but not the sample with the minimum clearance, the sample of the planned trajectory is considered in the tracking problem and it is assigned the weight $\mathbf{W}_1$, i.e.,

$$c(\mathbf{x}_n) < \bar{c} \quad \Longrightarrow \quad \mathbf{Q}_n = \mathbf{W}_1\,. \tag{12}$$

Third, if the clearance of a sample is not only below the threshold value but also the sample with the minimum clearance value, the sample of the planned trajectory is considered in the tracking problem and it is assigned the weight $\mathbf{W}_2$, i.e.,

$$\forall i \neq n,\ c(\mathbf{x}_n) < c(\mathbf{x}_i) \quad \Longrightarrow \quad \mathbf{Q}_n = \mathbf{W}_2\,. \tag{13}$$

In order to reach the goal state, the last sample of the planned trajectory must be part of the reference is weighted with $\mathbf{W}_\mathrm{G}$, i.e.,

$$n = N \quad \Longrightarrow \quad \mathbf{Q}_n = \mathbf{W}_\mathrm{G} \,. \tag{14}$$

The weighting matrix $\mathbf{Q}$ is assembled as a block-diagonal matrix of the respective samples' weights, i.e.,

$$\mathbf{Q} = \mathrm{blkdiag}\left(\mathbf{Q}_1 \; \mathbf{Q}_2 \; \ldots \; \mathbf{Q}_N\right) \,. \tag{15}$$

The weights $\mathbf{W}_1$ and $\mathbf{W}_2$ may depend on the clearance c and should chosen to reflect characteristics of the problem at hand.

Given the planned state trajectory and the weighting matrix $\mathbf{Q}$, we now employ iterative learning control to learn an input trajectory $\bar{\mathbf{u}}$ that tracks the planned state trajectory $\bar{\mathbf{x}}^*$ weighted by the matrix $\mathbf{Q}$ under the real-world dynamics. For this purpose, the so called lifted dynamics are given by

$$\bar{\mathbf{x}}_j = \mathbf{P}\bar{\mathbf{u}}_j + \mathbf{D}\mathbf{x}_\mathrm{I} \,, \tag{16}$$

where $\mathbf{P}$ is the matrix

$$\mathbf{P} := \begin{bmatrix} \mathbf{B} & \mathbf{0} & \ldots & \mathbf{0} \\ \mathbf{A}\mathbf{B} & \mathbf{B} & \ddots & \vdots \\ \vdots & \ddots & \ddots & \vdots \\ \mathbf{A}^{N-1}\mathbf{B} & \mathbf{A}^{N-2}\mathbf{B} & \ldots & \mathbf{B} \end{bmatrix} \tag{17}$$

and $\mathbf{D}$ is the matrix

$$\mathbf{D} := \left[\mathbf{A}^\top \quad \mathbf{A}^{2\top} \quad \ldots \quad \mathbf{A}^{N\top}\right]^\top \,. \tag{18}$$

On each trial, we first apply the current input trajectory $\bar{\mathbf{u}}_j$ to the real-world dynamics and record the corresponding state trajectory $\bar{\mathbf{x}}_j$. Next, we update the input trajectory by

$$\forall j \in \mathbb{N}, \quad \bar{\mathbf{u}}_{j+1} = \bar{\mathbf{u}}_j + \mathbf{\Delta}\bar{\mathbf{u}}_j . \tag{19}$$

To choose the increment $\mathbf{\Delta}\bar{\mathbf{u}} \in \mathbb{R}^{rN}$, we minimize the next-trial cost criterion

$$J_j(\mathbf{\Delta}\bar{\mathbf{u}}_j) = \hat{\mathbf{e}}_{j+1}^\top \mathbf{Q} \hat{\mathbf{e}}_{j+1} + \mathbf{\Delta}\bar{\mathbf{u}}_j^\top \mathbf{S} \mathbf{\Delta}\bar{\mathbf{u}}_j \,, \tag{20}$$

where $\hat{\mathbf{e}}_{j+1}$ is the difference between the planned and actual state trajectories predicted by the linear model on the next-trial, i.e.,

$$\hat{\mathbf{e}}_{j+1} = \bar{\mathbf{x}}^* - (\bar{\mathbf{x}}_j + \mathbf{P}\mathbf{\Delta}\bar{\mathbf{u}}_j) \,. \tag{21}$$

The optimization also takes constraints into consideration, i.e.,

$$\forall j \in \mathbb{N}, \quad \mathbf{\Delta}\bar{\mathbf{u}}_j = \underset{\mathbf{\Delta}\bar{\mathbf{u}}}{\mathrm{argmin}} \; J_j(\mathbf{\Delta}\bar{\mathbf{u}}) \tag{22}$$

$$\text{s.t.} \quad \bar{\mathbf{u}}_{j+1} \in \mathcal{U}, \; \bar{\mathbf{x}}_{j+1} \in \mathcal{X} \,. \tag{23}$$

The proposed scheme belongs to the class of so-called norm optimal ILC, for which monotonic convergence and asymptotic stability of the error trajectory over trials is guaranteed by design [13] if the plant matrix $\mathbf{P}$ is known.

Note that the kinodynamic RRT guarantees feasibility of the planned trajectory $\bar{\mathbf{x}}^*$ under the approximate, linear model (5). However, the trajectory may not be feasible under the real world's dynamics (2) if the difference between the approximate model and the actual dynamics is too large. In this case, the ILC would not converge to the desired trajectory $\bar{\mathbf{x}}^*$, the motion task may fail in the real world, and re-planning may be required, see Figure 2.

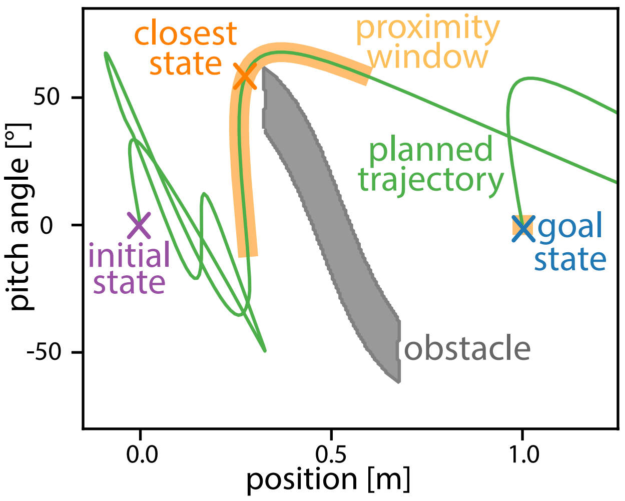

Fig. 3. The kinodynamic RRT finds a trajectory that connects the initial and goal state without colliding with the obstacle. Based on the distance to the obstacle, the samples of the state trajectory in the proximity window are weighted and selected to serve as reference samples for the point-to-point iterative learning control.

IV. Experimental Results

To validate the proposed method, we consider the problem of the two-wheeled inverted pendulum robot (TWIPR) that has to solve a motion task that consists in diving beneath an obstacle, see Figure 1. The robot consists of a chassis housing main electronics and to the chassis two motors with wheels are mounted such that the robot can balance in its upright position. The robots configuration is described by its pitch angle $\varphi \in \mathbb{R}$ and position $s \in \mathbb{R}$, i.e.,

$$\mathbf{y} = [\varphi \quad s]^\top \,, \tag{24}$$

and the robot's state vector consists of its pitch, position, and respective velocities, i.e.,

$$\mathbf{x} = [\dot{\varphi} \quad \varphi \quad \dot{s} \quad s]^\top \,. \tag{25}$$

The input variable is the motor torque $u \in \mathbb{R}$, and, hence, the robot is underactuated. A linear state feedback controller is implemented to stabilize and balance the robot in its upright equilibrium. The robots true dynamics are nonlinear and affected by complex phenomena such as friction and backlash of the gearings. However, only an approximate linear model of the dynamics has been identified.

In order to solve the given motion task, the proposed method is applied. Here, we first use the kinodynamic RRT to plan a state trajectory that connects the initial and goal state without any collisions or constraint violations. Planning is implemented in python using the Flexible Collision Checking library [14], and solving the planning problem takes roughly 8 s using a off-the-shelf notebook. The robot's configuration space, including the obstacle, and the planned trajectory is depicted in Figure 3. While the planned trajectory solves the problem, the trajectory includes excessive motion, and due to the model's inaccuracy, the input trajectory would not yield the same state trajectory if applied to the real robot.

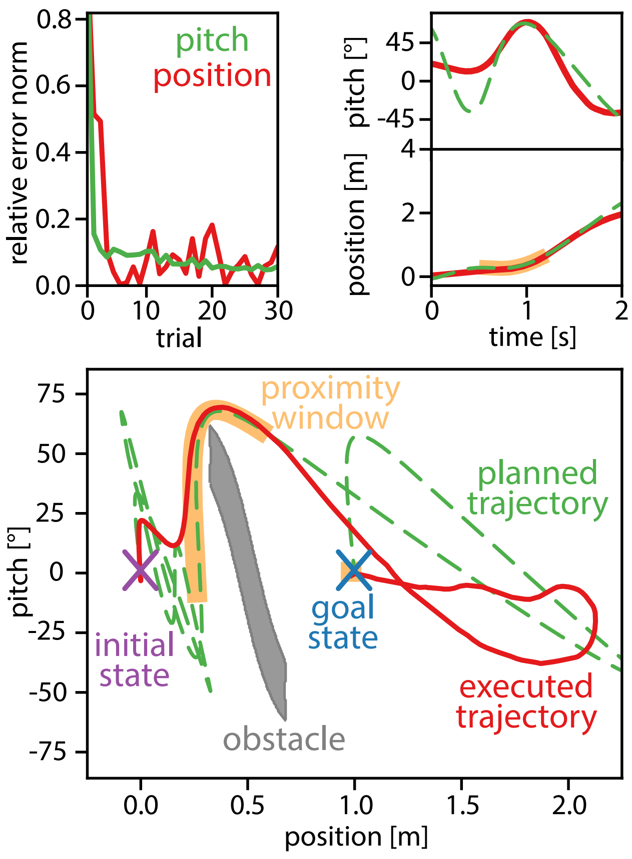

Fig. 4. The progression of the error norm shows that the iterative learning control learns to quickly track the desired reference samples under the real-world dynamics. The learned state trajectory not only solves the motion task by connecting the initial and goal state without a collision but also is rid of the planned trajectory's excessive motions.

Hence, the samples of the planned trajectory that are most relevant to the motion task are selected and weighted as reference points. The weights of the reference extraction are selected as

$$\mathbf{W}_1 = \mathrm{diag}(0 \quad 1 \quad 0 \quad 0)\frac{400}{1+\mathrm{e}^{-\alpha(c-c_{\mathrm{th}})}} \tag{26}$$

with $\alpha = 50$ and $c_{\mathrm{th}} = 0.12\,\mathrm{m}$,

$$\mathbf{W}_2 = \mathrm{diag}(0 \quad 0 \quad 200 \quad 4000) + \mathbf{W}_1\,, \tag{27}$$

and

$$\mathbf{W}_{\mathrm{G}} = \mathrm{diag}(100 \quad 100 \quad 100 \quad 100)\,. \tag{28}$$

Only samples that are sufficiently close to the obstacle are considered as reference samples, which are marked as proximity window in Figure 3.

To now solve the motion task in the real world, we employ ILC with the parameter $\mathbf{S} = \frac{1}{h^2}\mathrm{toeplitz}\left(-1 \quad 2 \quad -1\right)$ to learn to track the reference samples. Where h is the sampling period. A video of the learning process can be found at `www.bit.ly/3TC2tl3`, the results are depicted in Figure 4 and show that the ILC manages to rapidly decrease the tracking error which converges within roughly five trials to a value close to zero. The progression of the output trajectories over time shows that the planned trajectory is precisely tracked when in close proximity to the obstacle. The progression of the trajectories in configuration space shows that the precise tracking of the planned trajectory in the proximity window leads to the robot passing the obstacle without a collision. Furthermore, the trajectory not only avoids the obstacle but also ends in the goal state, i.e., the motion task was solved despite the robots nonlinear, underactuated dynamics.

V. Conclusion

This work has considered the problem of solving motion tasks for robots with challenging dynamics maneuvering environments that are obstructed by obstacles. A novel combination of methods from kinodynamic motion planning and iterative learning control was proposed. The proposed method only requires an approximate linear model even if the robot's actual dynamics are nonlinear and possibly underactuated. This is achieved by first employing rapidly exploring random trees to find a feasible solution to a motion task, and by then extracting and learning to track the most relevant samples of the planned trajectory using iterative learning control. The proposed method was validated on a two-wheeled inverted pendulum robot with underactuated dynamics that has to solve a motion task that requires a swift and agile motion. The experimental results indicate that the proposed method can enable robots to solve motion tasks in real-world environments despite incomplete model information and challenging dynamics.

One limitation of the proposed method is that the motion planning only employs an approximate model and can, hence, not guarantee feasibility of the motion under the real-world dynamics. To overcome this limitation, future work is going to employ Gaussian Processes in the ILC to learn a model of the unknown, nonlinear real-world dynamics [15] and use the learned model in the kinodynamic RRT to plan motions that are guaranteed to be feasible in the real-world.

Acknowledgment

Funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under Germany's Excellence Strategy - EXC 2002/1 "Science of Intelligence" - project number 390523135.

References

[1] G. Michalos, N. Kousi, S. Makris, and G. Chryssolouris, "Performance assessment of production systems with mobile robots," *Procedia CIRP*, vol. 41, pp. 195–200, 2016. Research and Innovation in Manufacturing: Key Enabling Technologies for the Factories of the Future - Proceedings of the 48th CIRP Conference on Manufacturing Systems.

[2] R. Murphy, "Activities of the rescue robots at the world trade center from 11-21 september 2001," *IEEE Robotics & Automation Magazine*, vol. 11, pp. 50–61, Sept. 2004.

[3] S. Xu, R. Zidek, Z. Cao, P. Lu, X. Wang, B. Li, and H. Peng, "System and experiments of model-driven motion planning and control for autonomous vehicles," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 52, no. 9, pp. 5975–5988, 2022.

[4] T. Seel, C. Werner, J. Raisch, and T. Schauer, "Iterative learning control of a drop foot neuroprosthesis—generating physiological foot motion in paretic gait by automatic feedback control," *Control Engineering Practice*, vol. 48, pp. 87–97, 2016.

[5] T. Marcucci, M. Petersen, D. von Wrangel, and R. Tedrake, “Motion planning around obstacles with convex optimization,” *arXiv preprint arXiv:2205.04422*, 2022.

[6] D. J. Webb and J. Van Den Berg, “Kinodynamic rrt*: Asymptotically optimal motion planning for robots with linear dynamics,” in *2013 IEEE international conference on robotics and automation*, pp. 5054–5061, IEEE, 2013.

[7] M. Vukosavljev, Z. Kroeze, A. P. Schoellig, and M. E. Broucke, “A modular framework for motion planning using safe-by-design motion primitives,” *IEEE Transactions on Robotics*, vol. 35, no. 5, pp. 1233–1252, 2019.

[8] M. Meindl, D. Lehmann, and T. Seel, “Bridging reinforcement learning and iterative learning control: Autonomous motion learning for unknown, nonlinear dynamics,” *Frontiers in Robotics and AI*, vol. 9, 2022.

[9] D. Bristow, M. Tharayil, and A. Alleyne, “A survey of iterative learning control,” *IEEE Control Systems Magazine*, vol. 26, no. 3, pp. 96–114, 2006.

[10] A. P. Schoellig, F. L. Mueller, and R. D’andrea, “Optimization-based iterative learning for precise quadrocopter trajectory tracking,” *Autonomous Robots*, vol. 33, pp. 103–127, 2012.

[11] D. Ronzani, J. Gillis, G. Pipeleers, and J. Swevers, “Ecoset-ilc: an iterative learning control approach with set-membership uncertainty,” in *2022 IEEE 17th International Conference on Advanced Motion Control (AMC)*, pp. 68–75, 2022.

[12] Y. Chen, B. Chu, and C. T. Freeman, “Point-to-point iterative learning control with optimal tracking time allocation,” *IEEE Transactions on Control Systems Technology*, vol. 26, no. 5, pp. 1685–1698, 2017.

[13] D. H. Owens and D. H. Owens, “Norm optimal iterative learning control,” *Iterative Learning Control: An Optimization Paradigm*, pp. 233–276, 2016.

[14] J. Pan, S. Chitta, and D. Manocha, “FCL: A general purpose library for collision and proximity queries,” in *2012 IEEE International Conference on Robotics and Automation*, pp. 3859–3866, 2012.

[15] M. Meindl, S. Bachhuber, and T. Seel, “Ai-mole: Autonomous iterative motion learning for unknown nonlinear dynamics with extensive experimental validation,” *Control Engineering Practice*, vol. 145, p. 105879, 2024.