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

DMP-based Path Planning for Model Predictive Interaction Control

Tim Goller, Daniel Brohm, Andreas Völz and Knut Graichen

Abstract— **This paper presents a three-layered hierarchical architecture to control manipulation tasks which involve interactions between the robot and workpieces. Learning from demonstration (LfD) is exploited to train dynamical movement primitives (DMPs) as fundamental building blocks for such tasks. Thus, position and wrench profiles are obtained from a kinesthetic demonstration, which makes the programming process intuitive for factory workers without expert knowledge. A model predictive path-following controller is used as the underlying control method, in order to make use of the explicit consideration of constraints within the controller formulation. Thereby, the progress parameter of the path-following control is used as the phase variable of the DMPs which results in a tight coupling of both methods. Finally, experimental results on a real robot system prove the effectiveness and real-time capable implementation of the approach.**

I. Introduction

Many industrial manufacturing tasks are nowadays still not completely automated but executed by human workers. One reason therefore is that most of the commercially available robotic systems require expert knowledge for creating workflow programs [1]. Learning from demonstration (LfD) or programming by demonstration (PBD) [2] address this problem with the aim of reducing the programming effort to a minimum and making this process as intuitive as possible.

Among the great variety of LfD approaches, dynamic movement primitives (DMPs) gained a lot of interest in the robotics community since they were developed in the early 2000s and later reformulated by Ijspeert et al. [3]. Besides their elegant formulation as a simple second-order attractor system, the possibility to model nonlinear discrete and rhythmic trajectories, contributed to their success. The major benefits are that the dynamic system, which encapsulates the nonlinearity, can be easily trained to fit a human demonstration, offers the possibility of temporal and spatial scaling in order to adapt to varying tasks and that stability can be investigated. For this reason, DMPs have been used to represent position trajectories in joint space [4], Cartesian space, force-torque profiles as well as combinations of both [5]–[7]. A comprehensive review about DMPs including the variety of formulations and their applications was recently published by Saverino et al. in [8].

More complex tasks require the execution of a sequence of motions and actions. With increasing complexity and length, these tasks cannot be implemented efficiently as single DMPs. Consequently, DMPs were integrated into task planning frameworks and combined with error recovery routines [9], [10] to execute them sequentially. In such frameworks, DMPs are used as the fundamental building blocks to generate trajectories for an underlying control scheme such as compliance control [6] or hybrid force/motion control [7]. With these approaches it is already possible to realize complex interactions as part of assembly tasks. Nevertheless, the consideration of constraints is difficult with these methods, especially if lifelong learning of the DMPs is implemented [8]. With model predictive control (MPC) [11] as an advanced control method, it is possible to consider constraints directly within the controller design. Because of this, several approaches for using MPC in robot control were developed [12]–[15] and tested on real hardware recently.

In previous work of the authors, an MPC scheme, namely the model predictive interaction control (MPIC), for controlling motions and interactions was introduced [14] and integrated into a hierarchical framework [16]. In the subsequent work [17], the MPIC was reformulated as a path-following controller in order to use the internal path dynamics to drive the progress along the task. However, the fundamental building blocks were implemented as classical trajectory or path planning primitives. In order to exploit the benefits of model predictive control and dynamic movement primitives in one framework, this paper presents an approach for combining them in a three-layered hierarchical control architecture, which integrates MPIC into an LfD approach. This results in the further advantage that smooth transitions of a DMP sequence are achieved by exploiting the predictive nature of the MPIC scheme and by using the path dynamics for variable temporal scaling.

II. Method

The considered system is a modern lightweight robot with N_{dof} degrees of freedom, which is capable of controlling interactions with its environment. Therefore, sensory information about the interaction wrench $\boldsymbol{\mathcal{F}}_{\mathrm{ext}}$ and a suitable control method are required. The approach presented in this paper relies on a three stage hierarchical control architecture as depicted in Fig. 1. As a basis, the robot is stabilized by a low-level PD controller with gravity compensation ( ). On the mid-level, a model predictive path-following controller computes the optimal desired joint positions and velocities for the underlying low-level loop and thus acts like an admittance controller. The Task-Skill-MP framework implements the abstract task planning on the top-level.

In the following section, the system dynamics is briefly summarized first, followed by the remaining components. However, the interested reader is referred to [17] for a detailed description of the path-following control and to [15] for a detailed derivation of the dynamic equations.

*This work was funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under project No GR 3870/5-1.

The authors are with the Chair of Automatic Control, Friedrich-Alexander-Universität Erlangen-Nürnberg (FAU), Erlangen, Germany. {tim.goller, daniel.brohm, andreas.voelz, knut.graichen}@fau.de

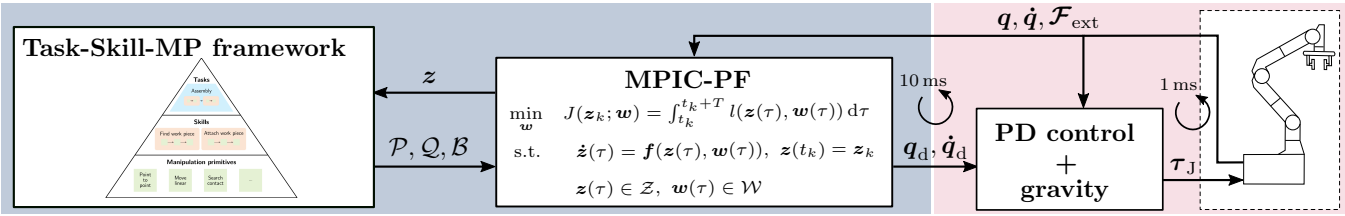

Fig. 1. System layout including the robot on the right ( ) and the online path planning on the left ( ). First, the task is demonstrated kinesthetically. Then a sequence of DMPs is trained to represent the desired motions and interaction wrenches acting at the end effector. During the execution phase, the sequence is evaluated by the Task-Skill-MP framework, nominal paths are generated and tracked by a model predictive path-following controller (MPIC-PF).

A. System Dynamics

The overall system dynamics consists of three individual subsystems which are the PD-controlled robot, the interaction between the end effector and its surrounding and a virtual dynamics of the path-following controller. Under the assumption that only sufficiently slow motions are realized, the closed loop dynamics of the PD-controlled robot can be simplified to a first-order system [15] with the actual and desired joint positions $\boldsymbol{q}, \boldsymbol{q}_\mathrm{d}$ as state variables. Moreover, the interaction wrench $\boldsymbol{\mathcal{F}}_\mathrm{ext}$ is considered as a state variable as well. Finally, a virtual second-order integrator with the progress s and velocity $\dot{s}$ along the path is used to realize the path-following control scheme [17]. This virtual subsystem is used to compute the current set point along the geometrical path in every sampling step. Together, these three subsystems compose the state

$$\boldsymbol{z} = [\boldsymbol{q}^\mathrm{T}, \boldsymbol{q}_\mathrm{d}^\mathrm{T}, \boldsymbol{\mathcal{F}}_\mathrm{ext}^\mathrm{T}, s, \dot{s}]^\mathrm{T} \in \mathbb{R}^{N_\mathrm{dof}+N_\mathrm{dof}+6+1+1} \tag{1}$$

and control input vector

$$\boldsymbol{w} = [\dot{\boldsymbol{q}}_\mathrm{d}^\mathrm{T}, \ddot{s}]^\mathrm{T} \in \mathbb{R}^{N_\mathrm{dof}+1} \tag{2}$$

with the desired joint velocities $\dot{\boldsymbol{q}}_d$ and the virtual acceleration along the path $\ddot{s}$. The Cartesian pose $\boldsymbol{p}$ is computed by evaluating the forward kinematics $\boldsymbol{p} = \mathrm{fk}(\boldsymbol{q})$ and is composed of the position $\boldsymbol{t} = [t_x, t_y, t_z]^\mathrm{T}$ and the quaternion $o = (\eta, \epsilon_x, \epsilon_y, \epsilon_z)$ for the orientation.

B. Task-Skill-MP Framework

Reducing the complexity of abstract manipulation tasks by hierarchical decomposition is a common approach in robotic applications [16], [18]–[21]. As a result, complex tasks are usually represented as a sequence of fundamental building blocks, so-called manipulation primitives (MPs). Due to their generic formulation, they can be reused and thus programming effort is reduced.

Based on the Task-Skill-MP framework [16], tasks are complex and abstract descriptions, skills are recurring subprocesses like peg-in-hole, push-button, etc. and implemented as a sequence of MPs. Manipulation primitives represent fundamental control strategies and are characterized by a generic formulation

$$\mathcal{MP} := (\mathcal{P}, \mathcal{Q}, \mathcal{B}, \mathcal{C})\,. \tag{3}$$

Therein, the control strategy $\mathcal{Q}$ is specified according to the control objective $\mathcal{P} \subset \{\boldsymbol{q}^*(s_\mathrm{loc}), \boldsymbol{p}^*(s_\mathrm{loc}), \boldsymbol{\mathcal{F}}^*(s_\mathrm{loc})\}$ with the MP-specific progress $s_\mathrm{loc} \in [0, 1]$. Additionally, system- or MP-specific constraints can be specified in $\mathcal{B}$ and the transitions in between the MPs are controlled by the conditions $\mathcal{C}$.

The generic definition (3) allows the user to implement a variety of primitives to realize motions, interactions and discrete actions like grasping. Furthermore, the control objective can be specified as a set point, geometric path or trajectory using arbitrary interpolation or planning methods. In previous work of the authors, trapezoidal trajectories [15] and piecewise linear paths [17] were used within the MPs. Another more advanced method to represent trajectories or paths based on human demonstrations are dynamic movement primitives [3], which are described in the next subsection.

C. Dynamic Movement Primitives

With DMPs [3], [8] it is possible to describe complex nonlinear trajectories for robotic systems. The essential core of a DMP is a linear spring damper system spanned between the actual and goal position y, y_g with a nonlinear forcing term $f(s_\mathrm{loc})$, i.e.

$$y''(s_\mathrm{loc}) = \alpha(\beta(y_\mathrm{g} - y(s_\mathrm{loc})) - y'(s_\mathrm{loc})) + f(s_\mathrm{loc})\,, \tag{4}$$

where $(\cdot)' = \frac{\mathrm{d}y}{\mathrm{d}s_\mathrm{loc}}$ and $(\cdot)'' = \frac{\mathrm{d}^2 y}{\mathrm{d}s_\mathrm{loc}^2}$ denote the derivatives with respect to the local MP-specific progress parameter s_loc.

The parameters α, β that define the dynamics of the spring damper system are mostly chosen such that critical damping is achieved. In the DMP formulation [3], [8], a temporal scaling factor τ is included, which is omitted in our approach because the actual time parameterization is considered as an additional degree of freedom and determined by the path-following controller. Thus, the time evolution of the MP-specific progress parameter $s_\mathrm{loc}(t)$ determines the temporal scaling.

The linear spring damper system of (4) can already be used for trajectory generation, yet only realizing linear motions. In order to achieve nonlinear profiles between the initial and goal position $y_0 = y(s_\mathrm{loc} = 0)$ and $y_\mathrm{g} = y(s_\mathrm{loc} = 1)$, a nonlinear forcing term

$$f(s_\mathrm{loc}) = \frac{\sum_{i=1}^{N_\mathrm{w}} w_i \Psi_i(s_\mathrm{loc})}{\sum_{i=1}^{N_\mathrm{w}} \Psi_i(s_\mathrm{loc})}(1 - s_\mathrm{loc})(y_\mathrm{g} - y_0) \tag{5}$$

is added with N_w weighted Gaussian kernel functions

$$\Psi_i(s_\mathrm{loc}) = \exp(-h_i((1 - s_\mathrm{loc}) - c_i)^2) \tag{6}$$

with width h_i and center location c_i along the phase variable. In the original DMP formulation [3], a phase variable is

defined that decays exponentially from 1 to 0 and is multiplied with f such that the forcing term vanishes at the goal position. Since in our approach $s_{\mathrm{loc}} \in [0,1]$ is increasing, the phase variable is represented by $(1 - s_{\mathrm{loc}})$. However, for the sake of simplicity, in the following s_{loc} is named the phase variable. The weights w_i are trained with locally weighted regression (LWR) [22] to fit the forcing term to the demonstrated trajectories.

By definition, a single DMP represents a one-dimensional trajectory. In order to describe multi-dimensional trajectories, N_{dmp} individual DMPs, i.e. one for each degree of freedom, are defined and synchronized by sharing the same phase variable. Thus, it is possible to define DMPs for joint positions, Cartesian pose and interaction wrench and to combine their outputs $y_i, i \in [1, \ldots, N_{\mathrm{dmp}}]$ to

$$\boldsymbol{y} = [y_1, \ldots, y_{N_{\mathrm{dmp}}}]^{\mathrm{T}}, \tag{7}$$

which is used as control objective for the MPIC. Note that discrete DMPs of the form (4) cannot be used for the orientation described by quaternions. In this case, another formulation is required as shown in [8], [23].

D. Model Predictive Interaction Control for Path-Following

The mid-level loop is the model predictive interaction control for path-following (MPIC-PF), which acts like an admittance control scheme in order to compute the desired joint positions and velocities for the low level PD-controller as shown in Fig. 1. The MPIC-PF is based on MPC [11], where the optimization problem

$$\min_{\boldsymbol{w}} \quad J(\boldsymbol{z}_k; \boldsymbol{w}; \mathcal{MP}_i; \mathcal{MP}_{i+1}) = \int_{t_k}^{t_k + T_{\mathrm{hor}}} l(\boldsymbol{z}(\tau), \boldsymbol{w}(\tau); \mathcal{MP}_i; \mathcal{MP}_{i+1})\, \mathrm{d}\tau \tag{8a}$$

$$\text{s.t.} \quad \dot{\boldsymbol{z}}(\tau) = f(\boldsymbol{z}(\tau), \boldsymbol{w}(\tau))\,,\ \boldsymbol{z}(t_k) = \boldsymbol{z}_k \tag{8b}$$

$$\boldsymbol{z}(\tau) \in \mathcal{Z},\ \boldsymbol{w}(\tau) \in \mathcal{W},\ \tau \in [t_k, t_k + T_{\mathrm{hor}}] \tag{8c}$$

is solved over a prediction horizon with length T and under consideration of the system dynamics (8b) and constraints (8c). Therein, the cost functional (8a) consists of the integral cost

$$l(\boldsymbol{z}(\tau), \boldsymbol{w}(\tau); \mathcal{MP}_i; \mathcal{MP}_{i+1}) = \psi(\tau) l_i(\boldsymbol{z}(\tau), \boldsymbol{w}(\tau), \mathcal{MP}_i) + (1 - \psi(\tau)) l_{i+1}(\boldsymbol{z}(\tau), \boldsymbol{w}(\tau), \mathcal{MP}_{i+1}) \tag{9}$$

with two cost parts l_i and l_{i+1} that depend on the system state $\boldsymbol{z}(\tau)$, the input $\boldsymbol{w}(\tau)$, and either on the current i-th or the subsequent $(i+1)$-th MP. Thus, the transition $\mathcal{MP}_{i \to i+1}$ is defined by the progress value s_{i+1} and is considered within the prediction by the activation function

$$\psi(\tau) = \begin{cases} 1 & \text{if } s(\tau) < s_{i+1} \\ 0 & \text{otherwise.} \end{cases} \tag{10}$$

A major benefit of the MPIC-PF is the possibility to implement various control strategies such as pure motion control in Cartesian or joint space, force control or hybrid forms by adjusting the controller parameterization during the MP transitions. The control strategy of the i-th MP is specified by the entries of the weighting matrices $\boldsymbol{Q}_q, \boldsymbol{Q}_p, \boldsymbol{Q}_{\mathcal{F}}$ of the integral cost

$$l_i(\boldsymbol{z}, \boldsymbol{w}; \mathcal{MP}_i) = \|\Delta \boldsymbol{q}\|^2_{\boldsymbol{Q}_q} + \|\Delta \boldsymbol{p}\|^2_{\boldsymbol{Q}_p} + \|\Delta \boldsymbol{\mathcal{F}}_{\mathrm{ext}}\|^2_{\boldsymbol{Q}_{\mathcal{F}}} + \|\boldsymbol{w}\|^2_{\boldsymbol{R}_w} + l_s(s, \dot{s}) \tag{11}$$

which scale the tracking error $\Delta \boldsymbol{x} = \boldsymbol{x} - \boldsymbol{x}^*$ of the respective quantity $\boldsymbol{x}$, i.e. joint positions $\boldsymbol{q}$, Cartesian pose $\boldsymbol{p}$ or interaction wrench $\boldsymbol{\mathcal{F}}_{\mathrm{ext}}$ and the control objective $\boldsymbol{x}^*$. Additionally, the control input is penalized by $\boldsymbol{R}_w$ and the path dynamics is considered in $l_s(s, \dot{s})$ to ensure a steady progress along the MP sequence [17].

E. MPC-based control of a MP sequence

As shown in Fig. 1, the MPIC-PF computes an optimal control signal with a cycle time of $10\,\mathrm{ms}$. Within this period, the optimization problem (8) is solved once over the prediction horizon T_{hor}. Thus, the DMPs are evaluated over the length of the prediction horizon to retrieve the nominal paths, depending on the time evolution of the progress parameter s. As the MPC horizon is shifted, this procedure of iterative path planning is repeated as shown in Algorithm 1.

In order to control a sequence of manipulation primitives, the progress parameter s as part of the state vector $\boldsymbol{z}$ is interpreted as a progress along the sequence with the value $s = 1$ marking the end of the execution. Thus, in every MP the current value of s is mapped to the local MP-specific progress by

$$s_{\mathrm{loc}}(\tau) = \frac{s(\tau) - s_i}{s_{i+1} - s_i}, \quad i = 0, \ldots, N_{\mathrm{MP}} - 1\,. \tag{12}$$

Then, all N_{dmp} DMPs are evaluated by calculating the forcing terms first and then solving (4) for the control objectives $y_j(s_{\mathrm{loc}})$, which are subsequently concatenated in the vector $\boldsymbol{y}(s_{\mathrm{loc}})$. Finally, the control objective $\boldsymbol{x}^* = \boldsymbol{y}(s_{\mathrm{loc}})$ is ready for use within the MPC cost function.

Algorithm 1 Iterative path planning

```
for τ ≤ T_hor do
    Calculate s_loc(τ) with (12)
    for j to N_dmp do                          ▷ DMP loop
        Calculate f_j(s_loc) using (5)
        Solve (4) for y_j(s_loc)
        Append y_j(s_loc) to y(s_loc)
        j ← j + 1
    end for
    Set x*(τ) = y(s_loc)
end for
```

III. Evaluation

For the experimental validation of the proposed approach, a Franka Emika robot with 7 degrees of freedom is used, which is equipped with a 6-axis force/torque sensor at the end effector and a handle bar for kinesthetic teaching. Note that it is important to measure the task-specific interaction wrench at the end effector since otherwise the human-caused guiding forces and torques (e.g. during free motions) would be included in the measurements. The complete experimental

Fig. 2. Experimental setup including the NIST task board #3 [24], a Franka Emika robot equipped with a 6-axis force/torque sensor and a handle bar for kinesthetic demonstration.

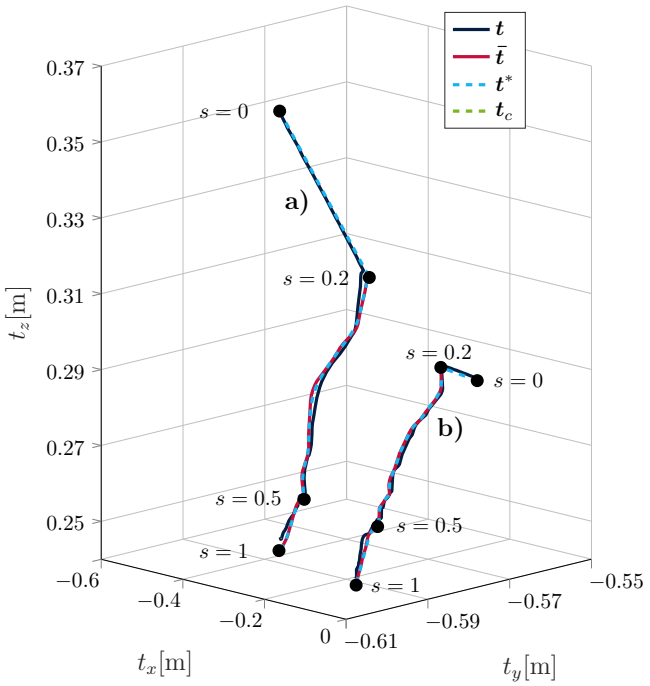

Fig. 3. Measurement results of the position profiles (—) for the snap assembly of an ethernet a) and an audio plug b). The corresponding demonstrations are plotted in blue (—) and the desired paths are drawn as dashed red lines (- -).

setup is depicted in Fig. 2. Therein, also a NIST task board #3 [24] is visible, which is used for snap assembly scenarios. Additionally, a peg-in-hole skill is considered for evaluating variations of the workpiece position. In addition to this paper, a video of all experiments can be found on the following link `https://www.fau.tv/clip/id/50427`.

A real-time capable implementation of the MPIC-PF is achieved with the MPC toolbox GRAMPC [25]. The controller runs on a desktop PC with Intel(R) Core(TM) i7-7700 CPU. As general GRAMPC settings, a prediction horizon of $T = 0.5\,\mathrm{s}$, $N_{\mathrm{hor}} = 40$ discretization steps, and $(i_{\mathrm{grad}}, i_{\mathrm{mult}}) = (4, 1)$ gradient and multiplier iterations are chosen. With these parameters an average and worst case computation time of $2.8\,\mathrm{ms}$ and $6.0\,\mathrm{ms}$ are achieved. The majority of the computational effort is caused by the calculation of the forward kinematics whereas the iterative evaluation and numerical integration of the DMPs can be neglected.

A. Snap Assembly

The NIST task board #3 is originally intended for evaluating a robot's capability to manipulate flexible wires [24]. However, it also offers three distinct connectors and the corresponding sockets. Among these three options, an ethernet and a $3.5\,\mathrm{mm}$ audio plug are chosen as snap assembly scenarios. Both are characterized by a force-sensitive insertion of the connector into the socket. Thereby, the required force profile is nonlinear due to the varying resistance which then causes the connector to snap into the socket.

As a first step, the complete snap assembly is demonstrated kinesthetically by a human. Then, the demonstration is divided manually into several intervals which require different control strategies such as motion control, force control or hybrid forms. Each interval is then considered as an individual MP. Automating this process is part of our ongoing work. The overall task is described by subsequently concatenating all intervals.

At the beginning of the task execution, i.e. in the interval $s \in [0, 0.2]$, the robot is moved to the initial position of the demonstration as depicted in Fig. 3. For this reason, no demonstrated position profile is plotted for this interval. In the second interval $s \in [0.2, 0.5]$, the DMPs generate a desired trajectory $\boldsymbol{t}^*$ (- -) from the demonstration $\bar{\boldsymbol{t}}$ (—). Since the end effector is not in contact with any environmental object, this interval is realized as a Cartesian motion control which is achieved by parameterizing the weighting matrices of the MPC cost functional (9) accordingly. The measured position profile is plotted in dark blue (—). Finally, the third MP implements a hybrid force/motion control for the connector insertion. Therein, the interaction force is controlled in the end effector's z-direction and the motion is controlled in the complementary directions. Fig. 4 shows the corresponding force profiles including again the demonstration $\bar{F}_z$ (—), desired force for the MPC F_z^* (- -), and the measured profile F_z (—). The areas highlighted gray denote the intervals where force control is active.

In the force profiles of Fig. 4 the nonlinearity in the demonstration is clearly visible between $s \approx 0.65$ and $s \approx 0.8$. Characteristic for both scenarios are the decreasing interaction force in the middle of the insertion process and the oscillations when the connector snaps into the socket. Regarding the oscillations, using DMPs for trajectory generation is beneficial since they automatically smoothen the noisy demonstration, cf. (- -) in Fig. 4.

The time evolution of the progress parameter s, which drives the ethernet snap assembly, is depicted in Fig. 5. It

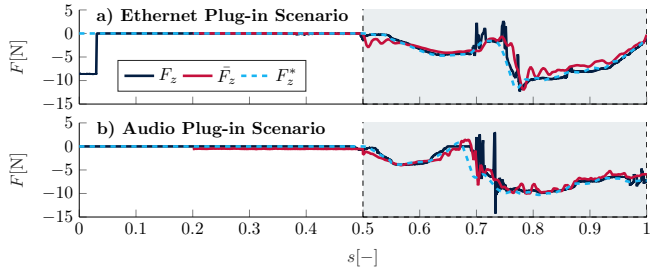

Fig. 4. Force profiles of both snap assembly scenarios with the demonstration (—), the desired force for the MPC (- -) and the measured forces of the executions (—).

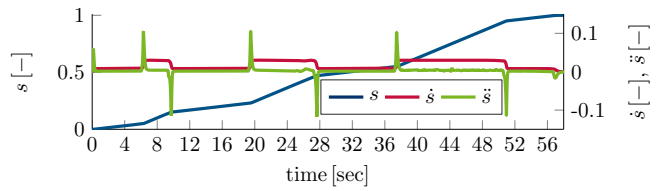

Fig. 5. Path dynamic of the ethernet snap assembly. The progress s is plotted in blue (—), the path velocity $\dot{s}$ is drawn in red (—) and the path acceleration $\ddot{s}$ is plotted in green (—).

can be seen that the progress velocity $\dot{s}$ is adjusted repeatedly to ensure smooth transitions between the MPs and to avoid jerky robot motions. This corresponds to the possibility of temporally scaling DMPs in every sampling step by the path dynamics, which is a major benefit of combining DMPs with the MPIC-PF.

B. Position Scaling with Constraints

In order to increase the efficiency and reusability in robot programming, it is desirable that the demonstrated workflow is applicable to varying workpiece positions. One major benefit of DMPs is that they can be spatially scaled by adjusting the goal position y_g in (4). By doing so, the qualitative nonlinear shape of the trajectory is not affected, but the spatial motion between the initial and goal position is adjusted. This can be seen in Fig. 6, where the position profiles of a peg-in-hole skill are depicted. Therein, the red curve (—) is the demonstrated motion. The curves to its left and right are spatially scaled and thus do not require a repeated demonstration. In the experiments, the position of the hole is assumed to be known, e.g. by using camera or laser systems for object tracking [26].

The difference to the snap assembly scenarios is that no force control is required during the insertion of the peg into the hole. Thus, the application is divided into four MPs. First, the robot is moved within $s \in [0, 0.2]$ to the initial position of the demonstration, which marks the actual start of the task, cf. Fig. 6. Then the end effector is approaching the hole and establishing contact within $s \in [0.2, 0.5]$. So far, both MPs are realized as Cartesian motion control. The following third MP implements the sliding motion towards the hole, such that the peg and the hole are aligned for insertion. Thus, in this interval $s \in [0.5, 0.7]$ hybrid force/motion control is needed. Therein, the force in the end effector's z-direction is controlled to maintain a stable contact and the motion is controlled in the complementary directions. Besides the

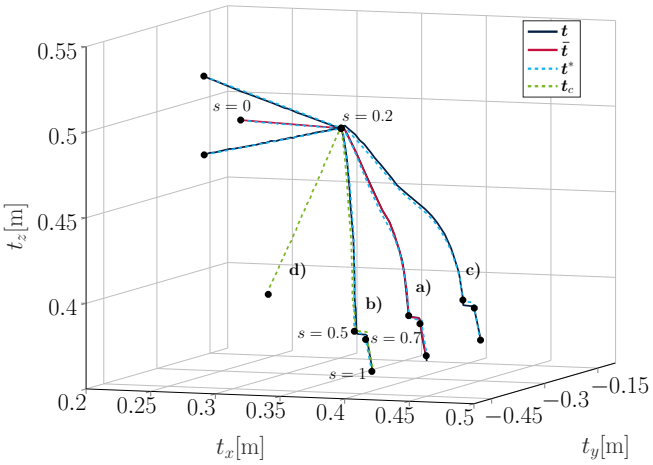

Fig. 6. Position profiles of the peg-in-hole skill for the unscaled a) and spatially scaled b), c) cases. The original demonstration is depicted in red (—), the desired positions for the MPC are drawn dashed blue (- -) and the measured profiles during the executions are plotted in dark blue (—). The profile of the constrained motion is drawn dashed green (- -) d).

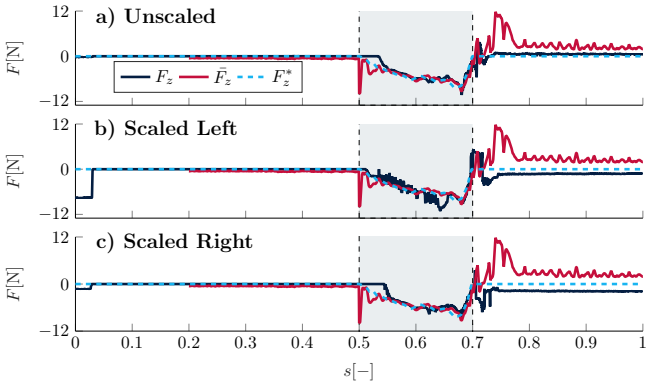

Fig. 7. Corresponding force profiles of the peg-in-hole skill variations. The original demonstration in all three subplots is the same and plotted in red (—). The desired force for the MPC is drawn dashed (- -) and the measured forces of the executions are plotted in dark blue (—).

motion (cf. Fig 6), the force profiles are depicted in Fig. 7. Therein, the gray areas denote the third MP where force control is applied. Finally, in the last MP $s \in [0.7, 1]$, the peg is inserted by a Cartesian motion control.

If motions and forces are demonstrated by hand guiding the robot through a workflow, it can be assumed that the resulting reference profiles (generated by the DMPs) are valid and do not lead to the robot exceeding system or task specific constraints. However, this is not the case anymore, if position scaling is applied to the DMPs. In such cases, the robot could potentially collide with environmental objects located in the workspace, approach its joint limits or singular configurations. However, those cases can be avoided by exploiting the redundancy of the robot and thus, by considering constraints for the motion execution. By using the MPIC-PF as the underlying control method, constraints can be defined system specific, e.g. minimum/maximum joint positions or task specific, e.g. Cartesian pose of the end effector or elbow. They are then considered directly within the controller formulation. Furthermore, by exploiting the Task-Skill-MP framework the constraints can be defined MP-

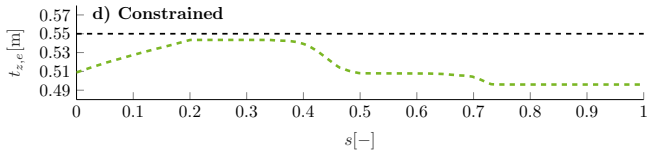

Fig. 8. Measured position profile of the elbow's z-coordinate for the constrained case of the scaled peg-in-hole task. The recorded profile is depicted in dashed green (- -) and the constraint in dashed black (- -)

specific and thus, changed during the execution phase. For avoiding a potential collision of the elbow with the ceiling due to scaling the motion to the left, the z-position of the elbow is now restricted to $t_{z,e} \in [0.1, 0.55]$. In Fig. 6 the resulting position profile of the end effector is depicted in dashed green (- -) for the constrained motion. Fig. 8 shows the constrained z-position of the elbow together with the maximum allowed position (- -). It can be seen, that the constraint is not violated while the desired pose of the end effector could be succesfully tracked.

IV. Conclusion

In this paper, an approach for combining dynamic movement primitives with model predictive interaction control is presented. The resulting system combines the benefits of both methods, which are the reduction of programming effort by using learning from demonstration, spatial and variable temporal scaling in every sampling step and the explicit consideration of constraints within the model predictive controller. Furthermore, a homogeneous controller architecture is achieved by using the MPIC-PF within the hierarchical Task-Skill-MP framework. Thus, complex manipulation tasks can be described as a sequence of DMPs which is subsequently executed by a single controller. Different control strategies such as joint or Cartesian motion control, force control or hybrid force/motion control are achieved by adjusting few parameters of the MPC cost functional. Additionally, constraints can be defined for each MP individually and are considered directly within the controller formulation.

The approach is validated on a real robot system which requires a real-time capable implementation of the model predictive controller. Three different force-sensitive scenarios show experimental results, including motion and force profiles. A video of the experiments can be found on the following link `https://www.fau.tv/clip/id/50427`.

Future work focuses on the online adaptation of task specific parameters which are difficult to demonstrate as, for example, a wiggling motion. This also includes the development of reliable error recovery strategies and a broader evaluation focusing on the robustness of the approach. Besides that, more complex tasks including gripper actions shall be realized in order to investigate the reusability of already demonstrated skills.

References

[1] O. Heimann and J. Guhl, "Industrial robot programming methods: A scoping review," in *Proc. of International Conference on Emerging Technologies and Factory Automation (ETFA)*, 2020, pp. 696–703.

[2] H. Ravichandar, A. S. Polydoros, S. Chernova, and A. Billard, "Recent advances in robot learning from demonstration," *Annual review of control, robotics, and autonomous systems*, vol. 3, pp. 297–330, 2020.

[3] A. J. Ijspeert, J. Nakanishi, H. Hoffmann, P. Pastor, and S. Schaal, "Dynamical movement primitives: learning attractor models for motor behaviors," *Neural computation*, vol. 25, no. 2, pp. 328–373, 2013.

[4] S. Schaal, J. Peters, J. Nakanishi, and A. Ijspeert, "Control, planning, learning, and imitation with dynamic movement primitives," in *IROS Workshop on Bilateral Paradigms on Humans and Humanoids*, 2003, pp. 1–21.

[5] P. Kormushev, S. Calinon, and D. G. Caldwell, "Imitation learning of positional and force skills demonstrated via kinesthetic teaching and haptic input," *Advanced Robotics*, vol. 25, no. 5, pp. 581–603, 2011.

[6] F. Steinmetz, A. Montebelli, and V. Kyrki, "Simultaneous kinesthetic teaching of positional and force requirements for sequential in-contact tasks," in *in Proc. of International Conference on Humanoid Robots (Humanoids)*, 2015, pp. 202–209.

[7] N. Wang, C. Chen, and A. Di Nuovo, "A framework of hybrid force/motion skills learning for robots," *IEEE Transactions on Cognitive and Developmental Systems*, vol. 13, no. 1, pp. 162–170, 2020.

[8] M. Saveriano, F. J. Abu-Dakka, A. Kramberger, and L. Peternel, "Dynamic movement primitives in robotics: A tutorial survey," *The International Journal of Robotics Research*, vol. 42, no. 13, pp. 1133–1184, 2023.

[9] F. J. Abu-Dakka *et al.*, "Solving peg-in-hole tasks by human demonstration and exception strategies," *Industrial Robot: An International Journal*, vol. 41, no. 6, pp. 575–584, 2014.

[10] N. Krüger *et al.*, "Technologies for the fast set-up of automated assembly processes," *KI-Künstliche Intelligenz*, vol. 28, pp. 305–313, 2014.

[11] E. F. Camacho and C. Bordons, *Model Predictive Control*. Springer, 2007.

[12] A. Wahrburg and K. Listmann, "MPC-based admittance control for robotic manipulators," in *in Proc. of Conference on Decision and Control (CDC)*, 2016, pp. 7548–7554.

[13] K. J. Kazim, J. Bethge, J. Matschek, and R. Findeisen, "Combined predictive path following and admittance control," in *in Proc. of Annual American Control Conference (ACC)*, 2018, pp. 3153–3158.

[14] T. Gold, A. Völz, and K. Graichen, "Model predictive interaction control for industrial robots," in *Proc. of IFAC World Congress*, 2020, pp. 10 026–10 033.

[15] ——, "Model predictive interaction control for robotic manipulation tasks," *IEEE Transactions on Robotics*, vol. 39, no. 1, pp. 76–89, 2022.

[16] T. Gold, A. Lomakin, T. Goller, A. Völz, and K. Graichen, "Towards a generic manipulation framework for robots based on model predictive interaction control," in *Proc. of International Conference on Mechatronics and Automation (ICMA)*, 2020, pp. 401–406.

[17] T. Goller, T. Gold, A. Völz, and K. Graichen, "Model predictive interaction control based on a path-following formulation," in *Proc. of International Conference on Mechatronics and Automation (ICMA)*, 2022, pp. 551–556.

[18] M. Pedersen *et al.*, "Robot skills for manufacturing: From concept to industrial deployment," *Robotics and Computer-Integrated Manufacturing*, vol. 37, pp. 282 – 291, 2016.

[19] B. Finkenmeyer, T. Kröger, and F. M. Wahl, "Executing assembly task specified by manipulation primitive nets," *Advanced Robotics*, vol. 19, no. 5, pp. 591–611, 2005.

[20] F. Steinmetz, V. Nitsch, and F. Stulp, "Intuitive task-level programming by demonstration through semantic skill recognition," *IEEE Robotics and Automation Letters*, vol. 4, no. 4, pp. 3742–3749, 2019.

[21] L. Johannsmeier, M. Gerchow, and S. Haddadin, "A framework for robot manipulation: Skill formalism, meta learning and adaptive control," in *Proc. of International Conference on Robotics and Automation (ICRA)*, 2019, pp. 5844–5850.

[22] C. G. Atkeson, A. W. Moore, and S. Schaal, "Locally weighted learning," *Artificial Intelligence Review*, vol. 11, pp. 11–73, 1997.

[23] F. J. Abu-Dakka, B. Nemec, J. A. Jørgensen, T. R. Savarimuthu, N. Krüger, and A. Ude, "Adaptation of manipulation skills in physical contact with the environment to reference force profiles," *Autonomous Robots*, vol. 39, pp. 199–217, 2015.

[24] K. Kimble *et al.*, "Benchmarking protocols for evaluating small parts robotic assembly systems," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 883–889, 2020.

[25] T. Englert, A. Völz, F. Mesmer, S. Rhein, and K. Graichen, "A software framework for embedded nonlinear model predictive control using a gradient-based augmented Lagrangian approach (GRAMPC)," *Optimization and Engineering*, vol. 20, no. 3, pp. 769–809, 2019.

[26] L. Pérez *et al.*, "Robot guidance using machine vision techniques in industrial environments: A comparative review," *Sensors*, vol. 16, no. 3, p. 335, 2016.