Final publication availabe at IEEEXplore http://dx.doi.org/10.1109/CCTA.2017.8062503

Mid-Level MPC and 6 DOF Output Path Following for Robotic Manipulators

Mathias Hauan Arbo*, Esten Ingar Grøtli† and Jan Tommy Gravdahl*
*Department of Engineering Cybernetics
NTNU, Norwegian University of Science and Technology
†Mathematics and Cybernetics, SINTEF DIGITAL, Trondheim, Norway

***Abstract*—In this article we discuss some of the benefits of using an MPC as a mid-level controller between the path generator and the low-level joint controller of a robot system. The MPC handles rudimentary runtime constraints that are not considered during path generation. We compare two task space oriented controllers: the model predictive path following controller and the model predictive trajectory tracking controller. We describe a 6 degrees of freedom reference path in terms of three points, and use this to experimentally verify the results with a UR5 robot and a UR3 robot.**

I. Introduction

Articulated industrial robots are used for a large variety of tasks which often entail moving the end-effector along a precomputed Cartesian path. When in enclosed spaces we can ensure that objects and obstructions are exactly known, but the real world is not so orderly. Objects may be moved around, obstructing the task at hand, motivating the use of controllers that can explicitly handle these observable constraints the obstructions impose. For some tasks obstructions can be handled during path generation, but when the path itself is pertinent, the obstructions should be considered during path execution.

The nonlinear model predictive controller (NMPC) views the control objective as an optimal control problem (OCP) with constraints and dynamics. Based on the predicted behavior of the system, it optimizes the input sequence so as to minimize an objective function. The first input in this sequence is applied to the actual system and the OCP is reformulated with new initial conditions based on the results of applying the input to the system. NMPCs allow us to handle rudimentary obstructions during path execution as constraints on the OCP.

There are two common schools in control of industrial manipulators. One considers the dynamics of the robot, where the torque applied at each joint as controllable. The other uses servo control on the robot joints, handling gravitational, Coriolis, friction and other forces internally, and exposes the end-user to kinematic references of the low-level controller. Kinematic control uses either position, velocity, or acceleration setpoints. The benefit of the kinematic approach is that it is simple and intuitive for end-users during free moving tasks. For fast-moving, high-inertia, or interaction tasks, dynamics play a greater role. In this article we will use joint velocity setpoints as the control interface, as this allows for application to different manipulators by redefining the forward kinematics.

We consider two MPC approaches: the model predictive path following controller (MPFC, following the convention of [1]), and the model predictive trajectory tracking controller (MPTTC). With MPTTC, the robot is to follow a path with an explicit path-timing. In [2], the path-timing is formulated as an OCP with constraints on the states. This is an open-loop approach that predefines a path-timing law for the trajectory tracking controller. In [3] this was extended to include constraints on the acceleration and inertial forces at the end-effector. For the MPFC, the path-timing dynamics are a part of the MPC, and are handled closed-loop. This allows the robot to move to minimize the deviation from the path, before moving along the path as will be demonstrated.

We will consider the output-path following MPFC of Faulwasser et al. [1]. This formulation focuses on paths defined in the output space. In [4], the MPFC is shown to converge to the path given appropriately chosen terminal constraints and penalties. In [5] the MPFC is used to generate torque inputs for a KUKA LWR IV robot. The OCP is solved using the ACADO framework [6], which uses sequential programming and the qpOASES active set solver. The robot is constrained to act as a two-link planar arm, and the path has 2 degrees-of-freedom (DOF).

The MPFC formulation is implemented for real-time contouring control of an x-y table in [7], where current commands are used to specify torques on the two servo drives. The dynamics are linearised and the OCP is formulated as a quadratic program which is solved using an active set solver. In [8] it is implemented for control of a toy tower crane. The tower crane is controlled with acceleration setpoints, and the OCP is solved using the gradient projection method which uses Pontryagin's maximum principle to solve the analytical OCP.

This article is a continuation of [9] which looks at the MPFC and MPTTC for a 2 DOF double pendulum system. It notes that Runge-Kutta gives a faster solver than collocation if the system is sufficiently simple, and shows how the MPFC can stop at obstructions that are not profitable, from the OCP perspective, to pass around. The MPTTC on the other hand will move along the nullspace of the constraint to follow the path.

In this article we extend the results of [5] and [9] to 6 DOF paths using a novel method of following 3 points, and argue for

the flexibility of output path following systems with kinematic control. We also provide experimental results of the 6 DOF path formulation for a spiral path with MPTTC and MPFC, and exemplify the flexibility by following a 3D Lissajous path with the UR3 robot and the UR5 robot.

II. Theory

A. Control

In Fig.1 we show a possible realization of the total control system. At the highest level a geometric path is precomputed based on the task to be performed, only considering static obstructions or task related obstructions. When the robot is to execute the task, the path is given to the mid-level MPC which uses cameras or similar systems to identify obstructions. The MPC gives kinematic setpoints to the fast low-level joint controllers on the robot. The output-path oriented control of the MPC means that we only need to generate the new forward kinematics for using it with a new robot, given that the interface to the low-level controller is similar. This flexible design means that tasks can be shared between different robot setups without the need for redesigning the path following and obstruction handling portion of the controller. The tasks can also be defined independent of the robots to be used, and a system can be devised which distributes ones available robots to the appropriate tasks.

B. Robot and Path

We consider a 6 DOF articulated robot, with $\boldsymbol{q} \in \mathbb{R}^6$ joint coordinates, and angular velocity setpoints $\boldsymbol{u} \in \mathbb{R}^6$.

Assumption 1. The low-level controllers are assumed to be sufficiently fast for

$$\dot{\boldsymbol{q}}(t) = \boldsymbol{u}(t) \tag{1}$$

to represent the robot dynamics.

Joint constraints are enforced as $\boldsymbol{q}(t) \in [\boldsymbol{q}_l, \boldsymbol{q}_u]$, and joint velocities are $\boldsymbol{u}(t) \in [\boldsymbol{u}_l, \boldsymbol{u}_u]$.

The base frame is located at the base of the robot, and the robot has known forward kinematics described using the Denavit-Hartenberg convention. The rotation from a reference frame situated at joint i to the base frame b is $\boldsymbol{R}_i^b \in \mathbb{R}^{3\times3}$ for $i = 1, \ldots, 6$, and $\boldsymbol{p}_{bi}^b$ is the coordinates of the reference frame i relative to frame b expressed in terms of frame b. The 6 DOF path is defined as $\boldsymbol{R}_d^b(s)$ and $\boldsymbol{p}_{bd}^b(s)$ defining rotation of the

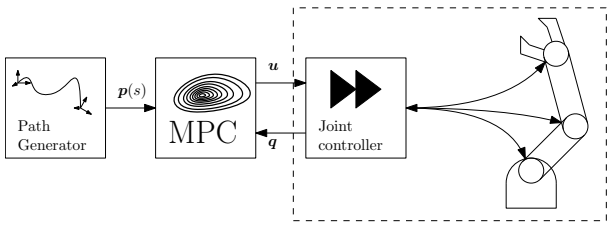

Fig. 1: Control hierarchy described in this article. The portion in the dashed box can be quickly changed if $\boldsymbol{u}$ and $\boldsymbol{q}$ are available through the same interface.

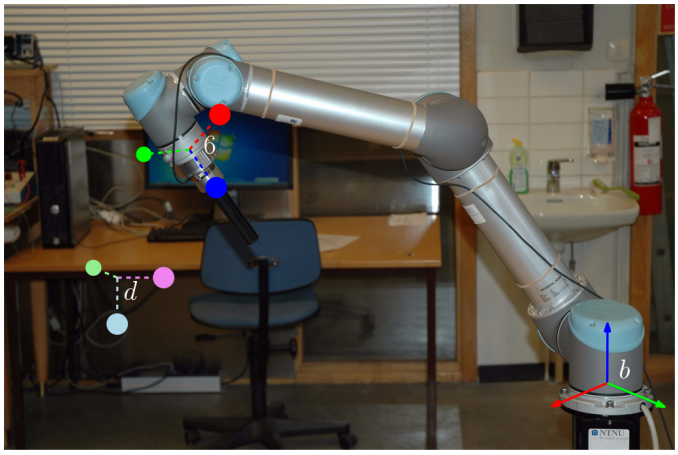

Fig. 2: The red, green, and blue points, attached to the UR5 follow the path of their lighter counterparts.

desired reference frame d w.r.t. the base frame, and s being the path-timing variable. We want the end-effector frame 6 to follow the desired reference frame d.

A variety of methods exist for representing rotations, e.g. quaternions, Euler angles, etc. We propose to use the intuitive method of defining three desired paths corresponding to orthogonal vectors from the desired reference frame:

$$\boldsymbol{p}(s) = \begin{bmatrix} \boldsymbol{p}_1(s) \\ \boldsymbol{p}_2(s) \\ \boldsymbol{p}_3(s) \end{bmatrix} = \begin{bmatrix} \boldsymbol{R}_d^b(s(t))[1,0,0]^T + \boldsymbol{p}_{bd}^b(s(t)) \\ \boldsymbol{R}_d^b(s(t))[0,1,0]^T + \boldsymbol{p}_{bd}^b(s(t)) \\ \boldsymbol{R}_d^b(s(t))[0,0,1]^T + \boldsymbol{p}_{bd}^b(s(t)) \end{bmatrix} \tag{2}$$

which three points in the end-effector frame are to follow. The points in the base frame are found using the forward kinematics as

$$\boldsymbol{h}(\boldsymbol{q}) = \begin{bmatrix} \boldsymbol{h}_1(\boldsymbol{q}) \\ \boldsymbol{h}_2(\boldsymbol{q}) \\ \boldsymbol{h}_3(\boldsymbol{q}) \end{bmatrix} = \begin{bmatrix} \boldsymbol{R}_{b6}^b(\boldsymbol{q})[1,0,0]^T + \boldsymbol{p}_{b6}^b(\boldsymbol{q}(t)) \\ \boldsymbol{R}_{b6}^b(\boldsymbol{q})[0,1,0]^T + \boldsymbol{p}_{b6}^b(\boldsymbol{q}(t)) \\ \boldsymbol{R}_{b6}^b(\boldsymbol{q})[0,0,1]^T + \boldsymbol{p}_{b6}^b(\boldsymbol{q}(t)) \end{bmatrix}, \tag{3}$$

where $\boldsymbol{h}_i(\boldsymbol{q})$ is the forward kinematics to point i.

This is visualized in Fig.2 as the red, green, and blue dots at the end-effector moving to the desired positions at their lighter counterparts.

We define the deviation from path as

$$\boldsymbol{e}_{\boldsymbol{p}}(t) := \boldsymbol{h}(\boldsymbol{q}(t)) - \boldsymbol{p}(s(t)) \tag{4}$$

with $\boldsymbol{e}_{\boldsymbol{p}}(t) \in \mathbb{R}^9$.

For the MPFC, we must also define the path-timing dynamics. We will use a simple double integrator

$$\ddot{s}(t) = v(t) \tag{5}$$

where $v(t) \in [-\infty, \infty]$ is the input, and $\dot{s}(t) \in [0, \dot{s}_u]$ is the non-negative path speed to ensure forward motion along the path. For more information on choice of path-timing dynamics we refer the reader to [4] and [8]. In theory and simulations we only required a single integrator, but in the experiments the double integrator path-dynamics performed better. We believe this was due to the delay caused by the solver making the system overshoot its desired path timing.

We will consider paths with a specific start and finish, $s \in [0, s_f]$ where s_f denotes the final value. We also define the deviation from the final value as

$$e_s(t) \coloneqq s(t) - s_f. \tag{6}$$

For the MPFC we define the augmented state vector $\boldsymbol{\xi} \coloneqq [\boldsymbol{q}, s, \dot{s}]^T$, augmented input $\boldsymbol{w} \coloneqq [\boldsymbol{u}, 0, v]^T$, augmented deviation $\boldsymbol{e_\xi} = [\boldsymbol{e_p}^T, e_s]^T$, and augmented system

$$\dot{\boldsymbol{\xi}}(t) = \boldsymbol{A}\boldsymbol{\xi}(t) + \boldsymbol{w}(t) \tag{7}$$

with

$$\boldsymbol{A} = \begin{bmatrix} \boldsymbol{0}_{6\times6} & \boldsymbol{0}_{6\times1} & \boldsymbol{0}_{6\times1} \\ \boldsymbol{0}_{1\times6} & 0 & 1 \\ \boldsymbol{0}_{1\times6} & 0 & 0 \end{bmatrix}. \tag{8}$$

C. Optimal Control Problem

Using the previously defined dynamics and deviations, we describe the OCP for the MPFC as

$$\min_{\boldsymbol{e_\xi}, \dot{\boldsymbol{e}}_{\boldsymbol{p}}, \boldsymbol{w}} \int_{t_k}^{t_k+T} J_{pf}(\tau, \bar{\boldsymbol{\xi}}(\tau), \bar{\boldsymbol{w}}(\tau))\mathrm{d}\tau \tag{9a}$$

s.t.:

$$\dot{\bar{\boldsymbol{\xi}}}(\tau) = \boldsymbol{A}\bar{\boldsymbol{\xi}}(\tau) + \bar{\boldsymbol{w}}(\tau) \tag{9b}$$
$$\bar{\boldsymbol{\xi}}(t_k) = \boldsymbol{\xi}(t_k) \tag{9c}$$
$$\bar{\boldsymbol{\xi}}(\tau) \in [\bar{\boldsymbol{\xi}}_l, \bar{\boldsymbol{\xi}}_u] \tag{9d}$$
$$\bar{\boldsymbol{w}}(\tau) \in [\bar{\boldsymbol{w}}_l, \bar{\boldsymbol{w}}_u] \tag{9e}$$
$$\boldsymbol{h}_c(\bar{\boldsymbol{\xi}}(\tau)) \leq 0 \tag{9f}$$

and for the MPTTC as

$$\min_{\boldsymbol{e_p}, \dot{\boldsymbol{e}}_{\boldsymbol{p}}, \boldsymbol{u}} \int_{t_k}^{t_k+T} J_{tt}(\tau, \bar{\boldsymbol{q}}(\tau), \bar{\boldsymbol{u}}(\tau))\mathrm{d}\tau \tag{10a}$$

s.t.:

$$\dot{\bar{\boldsymbol{q}}}(\tau) = \bar{\boldsymbol{u}}(\tau) \tag{10b}$$
$$\bar{\boldsymbol{q}}(t_k) = \boldsymbol{q}(t_k) \tag{10c}$$
$$\bar{\boldsymbol{q}}(\tau) \in [\bar{\boldsymbol{q}}_l, \bar{\boldsymbol{q}}_u] \tag{10d}$$
$$\bar{\boldsymbol{u}}(\tau) \in [\bar{\boldsymbol{u}}_l, \bar{\boldsymbol{u}}_u] \tag{10e}$$
$$\boldsymbol{h}_c(\bar{\boldsymbol{q}}(\tau) \leq 0 \tag{10f}$$

where the bar is to signify that these are internal states of the OCP, subscript u refers to the upper bounds, and subscript l refers to the lower bounds. The function $\boldsymbol{h}_c$ defines other constraints such as obstructions, the end-effector remaining in the workspace, etc. The OCP uses samples from time t_k and has a prediction horizon of length T.

The cost integrands are defined as

$$\begin{aligned} J_{pf}(\tau, \bar{\boldsymbol{\xi}}(\tau), \bar{\boldsymbol{w}}(\tau)) =& \frac{1}{2}\bar{\boldsymbol{e}}_{\boldsymbol{\xi}}(\tau)^T \boldsymbol{Q}_{\boldsymbol{\xi}} \bar{\boldsymbol{e}}_{\boldsymbol{\xi}}(\tau) \\ &+ \frac{1}{2}\dot{\bar{\boldsymbol{e}}}_{\boldsymbol{p}}(\tau)^T \boldsymbol{Q}_d \dot{\bar{\boldsymbol{e}}}_{\boldsymbol{p}}(\tau) \\ &+ \frac{1}{2}\bar{\boldsymbol{w}}(\tau)^T \boldsymbol{R}_{\boldsymbol{w}} \bar{\boldsymbol{w}}(\tau) \end{aligned} \tag{11}$$

fort the MPFC, and

$$\begin{aligned} J_{tt}(\tau, \bar{\boldsymbol{q}}(\tau), \bar{\boldsymbol{u}}(\tau)) =& \frac{1}{2}\bar{\boldsymbol{e}}_{\boldsymbol{p}}(\tau)^T \boldsymbol{Q}_{\boldsymbol{p}} \bar{\boldsymbol{e}}_{\boldsymbol{p}}(\tau) \\ &+ \frac{1}{2}\dot{\bar{\boldsymbol{e}}}_{\boldsymbol{p}}(\tau)^T \boldsymbol{Q}_d \dot{\bar{\boldsymbol{e}}}_{\boldsymbol{p}}(\tau) \\ &+ \frac{1}{2}\bar{\boldsymbol{u}}(\tau)^T \boldsymbol{R}_{\boldsymbol{u}} \bar{\boldsymbol{u}}(\tau) \end{aligned} \tag{12}$$

for the MPTTC. We have $\boldsymbol{Q}_{\boldsymbol{\xi}} = \mathrm{diag}(\boldsymbol{Q}_{\boldsymbol{p}}, q_s)$ and $\boldsymbol{R}_{\boldsymbol{w}} = \mathrm{diag}(\boldsymbol{R}_{\boldsymbol{u}}, r_v)$ with $\boldsymbol{Q}_{\boldsymbol{p}}$, $\boldsymbol{Q}_d$ and $\boldsymbol{R}_{\boldsymbol{u}}$ being positive definite, and scalars q_s and r_v positive.

D. Nonlinear Program

In this section we only give the discretization of (9) as the MPTTC is similar and simpler. We consider $\boldsymbol{u}(t)$ and $\boldsymbol{v}(t)$ to be piecewise continuous with time intervals of δ_t. The prediction horizon has $N_T = T/\delta_t$ intervals. This means that the prediction horizon is discretized from step t_k to t_{k+N_T}. Runge-Kutta of the 4th order (RK4) [10] gives

$$\bar{\boldsymbol{\xi}}_{k+1} = \bar{\boldsymbol{\xi}}_k + \frac{\delta_t}{6}(\boldsymbol{k}_1 + 2\boldsymbol{k}_2 + 2\boldsymbol{k}_3 + \boldsymbol{k}_4) \coloneqq \boldsymbol{F}(\bar{\boldsymbol{\xi}}_k, \bar{\boldsymbol{w}}_k) \tag{13}$$

with

$$\boldsymbol{k}_1 = \boldsymbol{A}\bar{\boldsymbol{\xi}}_k + \bar{\boldsymbol{w}}_k, \tag{14a}$$
$$\boldsymbol{k}_2 = \boldsymbol{A}\left(\bar{\boldsymbol{\xi}}_k + \frac{\delta_t}{2}\boldsymbol{k}_1\right) + \bar{\boldsymbol{w}}_k, \tag{14b}$$
$$\boldsymbol{k}_3 = \boldsymbol{A}\left(\bar{\boldsymbol{\xi}}_k + \frac{\delta_t}{2}\boldsymbol{k}_2\right) + \bar{\boldsymbol{w}}_k, \tag{14c}$$
$$\boldsymbol{k}_4 = \boldsymbol{A}\left(\bar{\boldsymbol{\xi}}_k + \delta_t\boldsymbol{k}_3\right) + \bar{\boldsymbol{w}}_k. \tag{14d}$$

We employ the simultaneous approach, and define the optimization vector as

$$\boldsymbol{x} = \begin{bmatrix} \bar{\boldsymbol{\xi}}_k^T & \bar{\boldsymbol{w}}_k^T & \dots & \bar{\boldsymbol{\xi}}_{k+N_T-1}^T & \bar{\boldsymbol{w}}_k^T & \bar{\boldsymbol{\xi}}_{k+N_T}^T \end{bmatrix} \tag{15}$$

where subscript k means that it is the discretised value of the state at time t_k. The dynamics and initial value are accounted for by

$$\boldsymbol{f}(\boldsymbol{x}) = \begin{bmatrix} \bar{\boldsymbol{\xi}}_k - \boldsymbol{\xi}(t_k) \\ \bar{\boldsymbol{\xi}}_{k+1} - F(\bar{\boldsymbol{\xi}}_k, \bar{\boldsymbol{w}}_k) \\ \vdots \\ \bar{\boldsymbol{\xi}}_{k+N_T} - F(\bar{\boldsymbol{\xi}}_{k+N_T-1}, \bar{\boldsymbol{w}}_{k+N_T-1}) \end{bmatrix} \tag{16}$$

and the constraints are accounted for by

$$\boldsymbol{f}_c(\boldsymbol{x}) = \begin{bmatrix} \bar{\boldsymbol{\xi}}_k - \bar{\boldsymbol{\xi}}_u \\ \bar{\boldsymbol{w}}_k - \bar{\boldsymbol{w}}_u \\ \vdots \\ \bar{\boldsymbol{\xi}}_{k+N_T} - \bar{\boldsymbol{\xi}}_u \\ \bar{\boldsymbol{\xi}}_l - \bar{\boldsymbol{\xi}}_k \\ \bar{\boldsymbol{w}}_l - \bar{\boldsymbol{w}}_k \\ \vdots \\ \bar{\boldsymbol{\xi}}_l - \bar{\boldsymbol{\xi}}_{k+N_T} \\ \boldsymbol{f}_c(\bar{\boldsymbol{\xi}}_k) \\ \vdots \\ \boldsymbol{f}_c(\bar{\boldsymbol{\xi}}_{k+N_T}) \end{bmatrix} \tag{17}$$

The resulting nonlinear program (NLP) is then

$$\min_{\boldsymbol{x}} \quad \phi(\boldsymbol{x}) \tag{18a}$$

s.t.:

$$\boldsymbol{f}(\boldsymbol{x}) = \boldsymbol{0} \tag{18b}$$

$$\boldsymbol{f}_c(\boldsymbol{x}) \leq \boldsymbol{0}, \tag{18c}$$

where the cost function is approximated with Euler's method

$$\phi(\boldsymbol{x}) = \sum_{j=k}^{k+N_T-1} \delta_t J_{pf}(t_j, \bar{\boldsymbol{\xi}}_j, \bar{\boldsymbol{w}}_j). \tag{19}$$

E. Interior point solver

Primal Interior point methods consider NLPs of the form

$$\min_{\tilde{\boldsymbol{x}}} \quad \phi(\tilde{\boldsymbol{x}}) - \mu \sum_{i=j}^{\tilde{n}} \ln(\tilde{\boldsymbol{x}}_i) \tag{20a}$$

s.t.:

$$\boldsymbol{f}(\tilde{\boldsymbol{x}}) = \boldsymbol{0} \tag{20b}$$

where $\tilde{\boldsymbol{x}}_i$ for $i < j$ are the previous optimization variables and $i < j$ are slack variables to make $\boldsymbol{f}_c$ an equality constraint. μ defines the steepness of the barrier associated with the slack variables. For large values of μ the $\ln$ term will dominate and the solution will tend to the middle of the feasible region. As μ decreases, ϕ will dominate and the solution will move towards the optimal solution. Solving (20) for decreasing μ will converge to the solution of the actual NLP (18).

The motivation for interior point solvers is that they have consistent runtime with respect to problem size, allowing us to potentially include more states and constraints without adversely affecting the runtime. They are however difficult to warm-start, as too low μ may make certain slack variables prematurely small and cause slow convergence. It was observed that warm-starting with the previously solved $\boldsymbol{x}$ gave a small decrease in runtime.

We will use the interior point solver IPOPT [11], a primal-dual interior point solver, solving (20) using the primal-dual equations, see Section 3.1 in [11]. Convergence of the MPFC can be ensured with terminal sets and penalties as in [4]. In this article we focus on run time and do not create terminal sets and penalties.

III. Experimental Results

In this section we describe the experiments performed. To compare the MPFC and MPTTC we use a UR5 that is to follow a spiral path. To illustrate the simplicity of using the same approach for different robots, we use a 3D Lissajous curve that is executed both by a UR3 and a UR5.

A. Implementation

The system was implemented using Python and the CasADi framework [12]. CasADi is a symbolic framework for defining optimization problems. It allows for: algorithmic differentiation, exploiting sparsity of the problem, and compiling the symbolic functions to C++ for faster execution. The framework supports a variety of solvers, both commercial and open-source. As of writing the fastest and most common solver is IPOPT [11].

We used the compilation flag "O2" to optimize the resulting functions. The experiments were performed on a Macbook Pro with a 2.5 Ghz i7 CPU. The timestep used in the simulations is $\delta_t = 0.05$, corresponding to an update rate of 20 Hz. The horizon has $N_T = 5$ timesteps corresponding to 0.25 s.

The forward kinematics were found using the Denavit-Hartenberg parameters described in [13]. The tuning parameters are given in Table I, and were the same for both the UR5 and the UR3.

B. Spiral path

In this section the reference path of the MPTTC and MPFC is a downward moving spiral path with a constant rotation of π rad around the y-axis from the base frame to the end-effector frame. The coordinate of the desired frame is

$$\boldsymbol{p}_{bd}^b(s) = \begin{bmatrix} 0.155\cos(2s) + 0.477 \\ 0.155\sin(2s) - 0.239 \\ 0.219 - 0.05s \end{bmatrix} \tag{21}$$

giving

$$\boldsymbol{p}(s) = \begin{bmatrix} \boldsymbol{p}_1(s) \\ \boldsymbol{p}_2(s) \\ \boldsymbol{p}_3(s) \end{bmatrix} = \begin{bmatrix} \boldsymbol{p}_{bd}^b(s) - [1,0,0]^T \\ \boldsymbol{p}_{bd}^b(s) + [0,1,0]^T \\ \boldsymbol{p}_{bd}^b(s) - [0,0,1]^T \end{bmatrix} \tag{22}$$

and the paths terminate at $s_f = 2\pi$. For the MPTTC we scale s by 0.0125 so that a full rotation is completed after approximately 80 s.

TABLE I: MPC Parameters

Parameter	$\boldsymbol{Q}_p$	$\boldsymbol{Q}_d$	$\boldsymbol{R}_u$	q_s	r_v
MPFC	$10^7\boldsymbol{I}_{9\times9}$	$1.5\cdot10^5\boldsymbol{I}_{9\times9}$	$10^{-4}\boldsymbol{I}_{6\times6}$	10^{-1}	10^{-4}
MPTTC	$10^7\boldsymbol{I}_{9\times9}$	$1.5\cdot10^5\boldsymbol{I}_{9\times9}$	$10^{-4}\boldsymbol{I}_{6\times6}$		

C. Spiral Results

In Fig.3 we see end-effector position of the MPFC following the described downward spiral. In Fig.4 we see the same for the MPTTC. Both controllers start a small distance from the start of the path. In Fig.5 we see the norms of $\boldsymbol{e_p}$ for the MPFC.In Fig.6 we see the norms of $\boldsymbol{e_p}$ for the MPTTC. Note that the MPFC converges faster than the MPTTC stemming from the MPFC first handling orientation and position before moving along the path. This is a trait of the path-following dynamics and can be adjusted by tuning q_s and r_v. The MPTTC on the other hand only has the positions of the desired points at each timestep, and will struggle to catch up with the desired orientation while also moving along the path.

It was observed that the run time of the solver depended on the configuration of the robot and deviation from the path. This is likely due to the solver entering local minima when solving. In Fig.7 we see the run times of the MPFC solver over time during the spiral path test. The run time is slightly longer before it has reached the path. When the path is reached, the

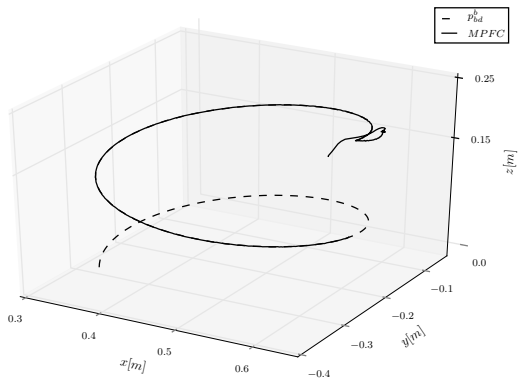

Fig. 3: The MPFC moves to the path before moving along it. The rotation to fit to the paths initial reference frame moves the origin a little off from the path until it converges.

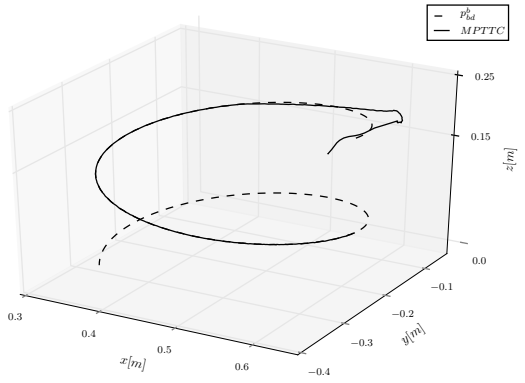

Fig. 4: The MPTTC moves along the path before matching the orientation. The deviation from the path is greater than for the MPFC.

run time stays relatively consistent. In Fig.8 the run times of the MPTTC is given. We see that the MPTTC solver struggles more than the MPFC when far from the path, but becomes more consistent when on the path.

Remark 1. In simulations $\dot{e}_s$ was not required in (11) or (12) for convergence. The delay caused by the solver and the robot interface made the proportional control result in the manipulator oscillating greatly around the path. The introduction of dampening along the path through $\dot{e}_s$ was fundamental for the implementation.

D. Robot Change

In Fig.10 we see the UR3 and UR5 moving to the 3D Lissajous path

$$\boldsymbol{p}^b_{bd}(s) = \begin{bmatrix} 0.035\cos(10s) + 0.1 \\ 0.035\cos(30s + 1) + 0.2 \\ 0.035\cos(20s + 1) + 0.3 \end{bmatrix} \tag{23}$$

which can be seen in Fig.9. The path was chosen to be as far from the home position $\boldsymbol{q}(0) = [0, -\pi/2, 0, -\pi/2, 0, 0]^T$, and

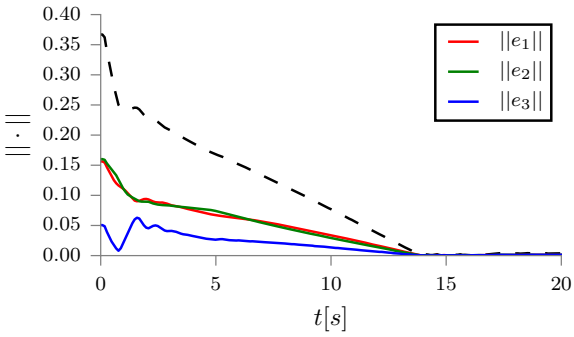

Fig. 5: Norm of the errors $\boldsymbol{e}_1$, $\boldsymbol{e}_2$ and $\boldsymbol{e}_3$ for the MPFC. The black stippled line is the sum of the norms.

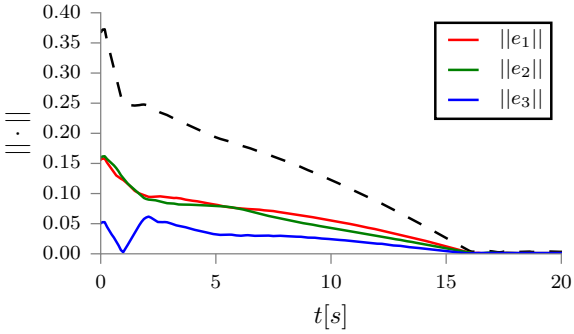

Fig. 6: Norm of the errors $\boldsymbol{e}_1$, $\boldsymbol{e}_2$ and $\boldsymbol{e}_3$ for the MPTTC. The black stippled line is the sum of the norms.

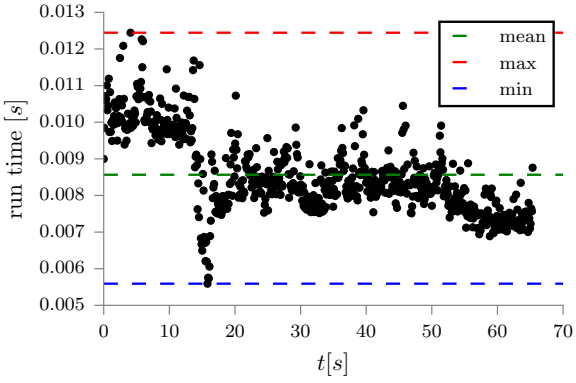

Fig. 7: Run times of the MPFC solver during execution of the path.

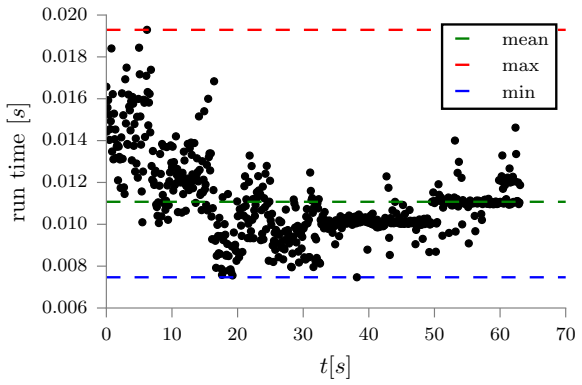

Fig. 8: Run times of the MPTTC solver during execution of the path.

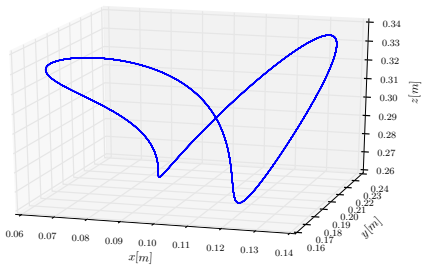

Fig. 9: The 3D Lissajous reference path is placed such that it is inside both the UR3 and the UR5 workspace.

small enough to be within the UR3's workspace. No change in tuning parameters was needed between the two robots. As the previous path, the desired rotation places the end-effector with its z-axis pointing downwards.

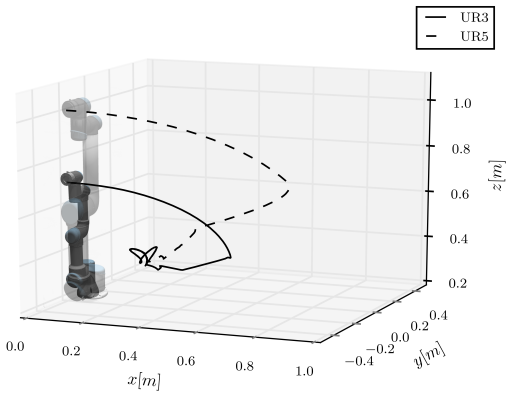

Fig. 10: The UR3 and UR5 start in their upright position, and move to the Lissajous path.

IV. Discussion and Future work

Both the MPFC and the MPTTC managed to follow the 6 DOF paths. The MPFC first moves to the initial path orientation before moving along the path. The MPTTC on the other hand has a moving setpoint that it must catch up with. The alternative to this is to have one controller to move the end-effector to the desired path and then switching to the along path controller. Using the MPFC allows us to tune the transition from approach to along path motion through the parameter q_s.

For a system where switching between active constraints does not happen often and rapidly, the interior point solver may not be the best option. Other solvers which benefit more from warm-starting may prove to give better run times. We believe the times when the solver takes longer than usual is a result of the restoration from local minima. We suggest testing with other solvers for evaluating the useability further. The longer run time of MPTTC stems from how it is implemented, the timing parameters s is included as a parameter, and we believe this may increase the run time slightly.

Delay in the interface and from the solver caused issues for the implementation of the system. The robot would only change joint velocity when a new command was sent, and as the solver could at times run longer than expected, there would be an integration error. This would cause the system to repeatedly overshoot, and oscillate around the desired position. With a faster implementation this is expected to be manageable, but dampening may still be desired.

V. Conclusion

Using MPC controllers as a mid-level controller between the path generator and the low-level controller allows us the flexibility of changing robots for the same task. Path generation is not necessarily the same as obstacle avoidance, and relinquishing that control to a system between the fast joint controller and the path generator may make for more flexible systems.

The definition of a 6 DOF path as three orthogonal points moving in space was useful for making the MPFC and MPTTC. We also experimentally demonstrated that the MPFC and the MPTTC were capable of following the desired paths.

VI. Acknowledgements

The work reported in this paper was based on activities within centre for research based innovation SFI Manufacturing in Norway, and is partially funded by the Research Council of Norway under contract number 237900.

References

[1] T. Faulwasser, B. Kern, and R. Findeisen, "Model predictive path-following for constrained nonlinear systems," *Proceedings of the 48h IEEE Conference on Decision and Control (CDC) held jointly with 2009 28th Chinese Control Conference*, no. 3, pp. 8642–8647, 2009.

[2] D. Verscheure, B. Demeulenaere, J. Swevers, J. De Schutter, and M. Diehl, "Time-optimal path tracking for robots: A convex optimization approach," *IEEE Transactions on Automatic Control*, vol. 54, no. 10, pp. 2318–2327, 2009.

[3] F. Debrouwere, W. Van Loock, G. Pipeleers, M. Diehl, J. Swevers, and J. De Schutter, "Convex time-optimal robot path following with Cartesian acceleration and inertial force and torque constraints," *Proceedings of the Institution of Mechanical Engineers, Part I: Journal of Systems and Control Engineering*, vol. 227, no. 10, pp. 724–732, nov 2013.

[4] T. Faulwasser and R. Findeisen, "Nonlinear Model Predictive Control for Constrained Output Path Following," *IEEE Transactions on Automatic Control*, vol. 9286, no. c, pp. 1–1, 2016.

[5] T. Faulwasser, T. Weber, P. Zometa, and R. Findeisen, "Implementation of Nonlinear Model Predictive Path-Following Control for an Industrial Robot," *IEEE Transactions on Control Systems Technology*, pp. 1–7, 2016.

[6] B. Houska, H. J. Ferreau, and M. Diehl, "ACADO Toolkit – An Open Source Framework for Automatic Control and Dynamic Optimization," *Optimal Control Applications and Methods*, vol. 32, no. 3, pp. 298–312, 2011.

[7] D. Lam, C. Manzie, and M. Good, "Application of model predictive contouring control to an X-Y table," *IFAC Proceedings Volumes (IFAC-PapersOnline)*, vol. 18, pp. 10 325–10 330, 2011.

[8] M. Böck and A. Kugi, "Real-time nonlinear model predictive path-following control of a laboratory tower crane," *IEEE Transactions on Control Systems Technology*, vol. 22, no. 4, pp. 1461–1473, 2014.

[9] M. H. Arbo, E. I. Grøtli, and J. T. Gravdahl, "On Model Predictive Path Following and Trajectory Tracking for Industrial Robots," in *13th IEEE Conference on Automation Science and Engineering (CASE)*, 2017.

[10] O. Egeland and J. T. Gravdahl, *Modeling and Simulation for Automatic Control*. Trondheim: Marine Cybernetics, 2003.

[11] A. Wächter and L. T. Biegler, *On the Implementation of Primal-Dual Interior Point Filter Line Search Algorithm for Large-Scale Nonlinear Programming*, 2006, vol. 106, no. 1.

[12] J. Andersson, "A General-Purpose Software Framework for Dynamic Optimization," PhD thesis, Arenberg Doctoral School, KU Leuven, Belgium, 2013.

[13] "Actual center of mass for robot - 17264 | Universal Robots." [Online]. Available: https://www.universal-robots.com/how-tos-and-faqs/faq/ur-faq/actual-center-of-mass-for-robot-17264/