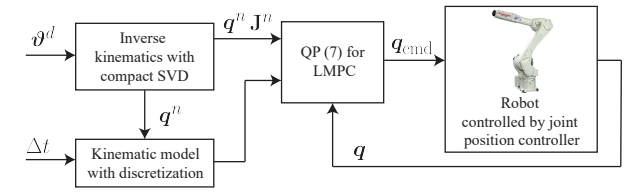
2023 IEEE International Conference on Robotics and Automation (ICRA 2023)
May 29 - June 2, 2023. London, UK

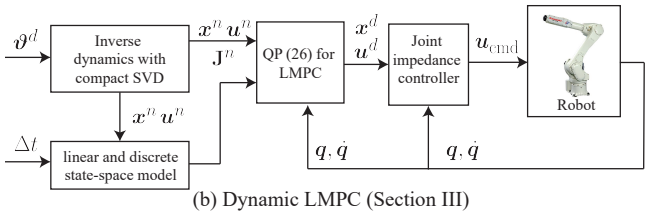
Real-Time Model Predictive Control for Industrial Manipulators with Singularity-Tolerant Hierarchical Task Control

Jaemin Lee[1], Mingyo Seo[2], Andrew Bylard[3], Robert Sun[3], and Luis Sentis[2]

***Abstract*— This paper proposes a real-time model predictive control (MPC) strategy for accomplishing multiple tasks using robots within a finite-time horizon. In industrial robotic applications, it is crucial to consider various constraints to ensure that joint position, velocity, and torque limits are not exceeded. In addition, singularity-free and smooth motions require executing tasks continuously and safely. Instead of formulating nonlinear MPC problems, we devise linear MPC problems using kinematic and dynamic models linearized along nominal trajectories produced by hierarchical controllers. These linear MPC problems are solvable via the use of Quadratic Programming; therefore, we significantly reduce the computation time of the proposed MPC framework so the resulting update frequency is higher than 1 kHz. Our proposed MPC framework is more efficient in reducing task tracking errors than a baseline based on operational space control (OSC). We validate our approach in numerical simulations and in real experiments using an industrial manipulator. More specifically, we deploy our method in two practical scenarios for robotic logistics: 1) controlling a robot carrying heavy payloads while accounting for torque limits, and 2) controlling the end-effector while avoiding singularities.**



(a) Kinematic LMPC (Section II)



(b) Dynamic LMPC (Section III)

Fig. 1: **Block diagrams of the proposed MPC approaches**: (a) Kinematic MPC for robots controlled by a joint position controller, (b) Dynamic MPC for torque-controlled robots.

I. Introduction

Robotic systems have been broadly utilized in automated industrial applications such as logistics. One of the critical concerns in these environments is the safety and success of manipulation tasks, such as packing, palletizing, and depalletizing [8]. To perform the above tasks, robotic manipulators must perform swift and secure operation to grab, manipulate, and toss boxes, objects, or parcels. These often require careful specification and tracking of task-space motions and hierarchical task-space objectives, for example to prioritize position control while maintaining sensible payload orientations throughout a trajectory. Carrying heavy payloads is also critical for robotic manipulators in terms of both their mechanical design [2] and their controller implementation [1]. In this paper, we aim to improve the task motion tracking performance of robots while generating fast, safe and smooth motions.

Operational Space Control (OSC) and Task Space Control (TSC) have been employed to generate dynamically consistent torque commands to effectively and safely track task trajectories [13, 21, 23]. For instance, OSC has proven effective in improving the tracking performance of heavy manipulators by considering feedforward terms based on robot dynamics [19]. In addition, robot manipulators can generate compliant behaviors for safe manipulation by using TSC with null-space projection matrices in the presence of humans [26]. Many advanced approaches based on OSC and TSC allow manipulators to safely move in the vicinity of singularities [12, 16] or consider torque input saturation in operational space [4, 22]. However, OSC and TSC are based on single-step optimizations resulting on myopic motions (only optimal locally at each time step) and are susceptible to abrupt changes due to the effect of the task specifications or constraints.

Model predictive control (MPC) has been frequently combined with impedance control [3] or inverse dynamics control [10, 24] to control manipulators. Usually, the feedback MPC is updated at lower frequency (20 Hz - 50 Hz), compared with the feedback control frequency (400 Hz - 1 kHz) [9, 15]. An ideal MPC should execute with direct sensor feedback at a high-frequency update rate (500 Hz - 1 kHz) [14]. However, the nonlinear problem formulation resulting from the dynamic models and constraint requirements imposed by robotic manipulators makes it difficult to solve these problems at a fast rate using MPC, especially when coupled with hierarchical task specifications, which are increasingly in demand in industrial applications.

MPC problems have previously been formulated as quadratically-constrained quadratic programs [17] and sequential optimization problems [27], then solved via convex optimization. However, the speed of these MPC approaches are still not sufficient for implementation within a real-time

We thank Dexterity and the HCRL personnel. Jaemin Lee and Mingyo Seo were interns for Dexterity Inc. during the summer of 2022. Also, Luis Sentis was a consultant for Dexterity Inc. during the summer of 2022.

[1]J. Lee is with the Department of Mechanical and Civil Engineering, California Institute of Technology, Pasadena, CA, USA, jaemin87@caltech.edu

[2]M. Seo and L. Sentis are with The University of Texas at Austin, Austin, TX, USA, {mingyo, lsentis}@utexas.edu

[3]A. Bylard, and R. Sun are with Dexterity Inc., Redwood City, CA, USA, {andrew.bylard, robert}@dexterity.ai

control loop to provide the needed industrial performance. For industrial applications, it is extremely important to implement the hierarchical task control while improving the task tracking performance via MPC within a real-time control loop. Recent studies have tried efficient replacements for the nonlinear models [25] or learning the terminal cost [11] using data-driven techniques such as neural networks increasing their task tracking performance. However, they are complicated to implement, they produce limited reduction of computation time, and they are missing key details of their computational performance such as dependence on the receding horizon and their control computer specifications.

Three significant issues arise when employing MPC in industrial manipulation: (1) dealing with nonlinear dynamics and cost functions, (2) dealing with hierarchical task specifications, and (3) guaranteeing stability and robustness to singularities while tracking task trajectories. To resolve the above issues, we aim to formulate kinematic and dynamic MPC approaches which can be executed with low-level controllers at a high-frequency update rate, as shown in Fig. 1. Our framework uses inverse kinematic and dynamic control formulations in the operational space to generate nominal trajectories. Since the stability of OSC while performing hierarchical tasks is verified in [7], it is assumed that using OSC provides stable nominal trajectories as inputs for the proposed MPC. In addition, we enforce terminal state constraints to stabilize our MPC [20]. Based on the input nominal trajectories, we devise a simple formulation for MPC to reduce task tracking errors with additive cost terms for generating smooth motions. Our kinematic and dynamic MPC approaches are formulated as Quadratic Programming (QP) problems, which can be solved very fast.

The main contributions of our work are as follows. First, we propose a framework that integrates hierarchical control, MPC, and low-level control. In particular, our proposed MPC framework, which is for executing multiple tasks with hierarchy, is updated at the same fast rate as the low-level control loop. We verify that the proposed MPC is sufficiently fast to be executed at a 1 kHz closed-loop update rate through both experiments and simulations. We also analyze the effect of the receding horizon on computation time. Second, we report that our MPC results follow the task priority imposed by the hierarchical controller while reducing the task tracking error. We showcase simulations showing that the proposed MPC framework reduces task tracking errors when performing manipulation tasks with a heavy payload under torque saturation. Third, our proposed MPC framework helps to avoid the involuntary termination of robot movement when the joint velocities exceed their limit when passing through singularities. We experimentally show that the proposed MPC generates stable and smooth behaviors in the vicinity of the singularities. For validation, we apply the proposed MPC framework to two industrial Kawasaki manipulators: the RS007N and RS020N models.

The remainder of this paper is organized as follows. We propose our kinematic MPC and explain its implementation in Section II. Section III presents the proposed dynamic MPC with linearization and discretization in detail. In Section IV, we deploy the proposed MPC approaches to demonstrate two practical scenarios using industrial manipulators. Numerical simulations and experiments show that the proposed MPC approach reduces tracking errors and generates safe smooth motions.

II. Kinematic MPC with Constraints

A. Kinematic Model with Discretization

We perform simple discretization of the joint velocity and acceleration with a time interval Δt (usually identical to the time interval for updating a low-level controller) as follows:

$$\dot{\boldsymbol{q}}_i = \frac{\boldsymbol{q}_i - \boldsymbol{q}_{i-1}}{\Delta t}, \quad \ddot{\boldsymbol{q}}_i = \frac{\boldsymbol{q}_i - 2\boldsymbol{q}_{i-1} + \boldsymbol{q}_{i-2}}{\Delta t^2}. \tag{1}$$

Now, let us consider a task function $g : \mathbb{R}^n \to \mathbb{R}^{n_g}$ which is C^1 and $\boldsymbol{\vartheta} = g(\boldsymbol{q})$ with

$$\Delta\boldsymbol{\vartheta} = \mathbf{J}_k(\boldsymbol{q})\Delta\boldsymbol{q}, \tag{2}$$

where $\mathbf{J}(\boldsymbol{q}) \in \mathbb{R}^{n_g \times n}$. For instance, the functional output of this task can be the position and orientation of the end-effector, represented by $\boldsymbol{\vartheta}$. Then, the low-level controller will drive the robot to track the desired joint position trajectory. Given a finite-time horizon $\mathcal{T} = [t_i,\ t_f]$ where $t_i < t_f$, we vertically concatenate the joint positions/velocities/accelerations such that $\hat{\boldsymbol{q}} = [\boldsymbol{q}_i^\top, \cdots, \boldsymbol{q}_f^\top]^\top$, $\hat{\dot{\boldsymbol{q}}} = [\dot{\boldsymbol{q}}_i^\top, \cdots, \dot{\boldsymbol{q}}_f^\top]^\top$, and $\hat{\ddot{\boldsymbol{q}}} = [\ddot{\boldsymbol{q}}_i^\top, \cdots, \ddot{\boldsymbol{q}}_f^\top]^\top$. Then, the joint velocity and acceleration in (1) are expressed as follows:

$$\hat{\dot{\boldsymbol{q}}} = \mathbf{S}_v\hat{\boldsymbol{q}} + \boldsymbol{v}, \quad \hat{\ddot{\boldsymbol{q}}} = \mathbf{S}_a\boldsymbol{q} + \boldsymbol{a}, \tag{3}$$

where $\mathbf{S}_v$ and $\mathbf{S}_a$ are constant matrices. Assuming that the initial velocity and acceleration are zero, we can obtain components of the vectors $\boldsymbol{v}$ and $\boldsymbol{a}$ as constant vectors. As a result, we express the joint velocity and acceleration in linear affine form in terms of the joint position.

B. MPC formulation

We aim to formulate a generic optimization problem for our kinematic MPC approach. Compared to OSC, our kinematic MPC generates smoother and more robust control commands by tightly integrating them with a low-level joint position controller. This section assumes that the low-level joint position controller is well-tuned and stable within the desired motion bandwidth.

Given a task trajectory $(\boldsymbol{\vartheta}_0^d, \cdots, \boldsymbol{\vartheta}_N^d)$, the optimal control problem is formulated with a prediction horizon n_p. The cost function consists of task tracking errors, joint damping, and joint acceleration terms. In addition, the joint position and velocity limits are considered in the optimization problem:

Continuous-Kinematic-MPC:

$$\begin{aligned} \min_{\hat{\boldsymbol{q}}} \quad & \mathcal{J} = \sum_{k=i}^{i+n_p} \Big((\boldsymbol{\vartheta}_k^d - g(\boldsymbol{q}_k))^\top \mathbf{Q}_e (\boldsymbol{\vartheta}_k^d - g(\boldsymbol{q}_k)) \\ & \qquad + \dot{\boldsymbol{q}}_k^\top \mathbf{Q}_d \dot{\boldsymbol{q}}_k + \ddot{\boldsymbol{q}}_k^\top \mathbf{Q}_a \ddot{\boldsymbol{q}}_k \Big), \\ \text{s.t.} \quad & \dot{\boldsymbol{q}}_{\min} \le \dot{\boldsymbol{q}}_k \le \dot{\boldsymbol{q}}_{\max}, \\ & \boldsymbol{q}_{\min} \le \boldsymbol{q}_k \le \boldsymbol{q}_{\max}, \quad \forall k \in \{i, \cdots, i+n_p\}, \end{aligned} \tag{4}$$

where $\mathbf{Q}_e \in \mathbb{R}^{n_g \times n_g}$, $\mathbf{Q}_d \in \mathbb{R}^{n \times n}$, and $\mathbf{Q}_a \in \mathbb{R}^{n \times n}$ denote the weighting matrices for the tracking task error, joint damping, and joint acceleration terms, respectively. Since the task mapping g is a nonlinear functional mapping, the first

term in the cost function, $(\boldsymbol{\vartheta}_k^d - g(\boldsymbol{q}_k))^\top \mathbf{Q}_e (\boldsymbol{\vartheta}_k^d - g(\boldsymbol{q}_k))$, is not convex in terms of $\boldsymbol{q}_k$. Therefore, in the current formulation in (4), it is not possible to solve directly via QP due to the task tracking error term in the cost function.

To reformulate the above MPC problem as a QP problem, we linearize the nonlinear term along the nominal trajectory. By only considering the first term in the Taylor expansion, we approximate the nonlinear term as follows:

$$\boldsymbol{\vartheta}_k^d - g(\boldsymbol{q}_k) \approx \mathbf{J}_k^n (\boldsymbol{q}_k^n - \boldsymbol{q}_k), \tag{5}$$

where $\mathbf{J}_k^n = \mathbf{J}(\boldsymbol{q}_k^n)$ and $\boldsymbol{q}^n$ represents a nominal joint trajectory obtained by solving the inverse kinematics problem. We consider $\mathbf{J}_k^n$ as a stack of projected Jacobian matrices for n_t tasks such as $\mathbf{J}_k^{n,(j|\text{prev})}$ where $j \in \{1, \cdots, n_t\}$ in (8).

In turn, we convexify the cost function by using the above approximation and the linear models for the joint velocity and acceleration:

$$\begin{aligned}\mathcal{J} \approx & \left(\sum_{k=i}^{i+n_p} \boldsymbol{q}_k^\top \mathbf{J}_k^{n\top} \mathbf{Q}_e \mathbf{J}_k^n \boldsymbol{q}_k - 2 \boldsymbol{q}_k^\top \mathbf{J}_k^n \mathbf{Q}_e \boldsymbol{q}_k \right) + \\ & \hat{\boldsymbol{q}}^\top \left(\mathbf{S}_v \hat{\mathbf{Q}}_d \mathbf{S}_v + \mathbf{S}_a \hat{\mathbf{Q}}_a \mathbf{S}_a \right) \hat{\boldsymbol{q}} + 2 \left(\boldsymbol{v}^\top \hat{\mathbf{Q}}_d \mathbf{S}_v + \boldsymbol{a}^\top \hat{\mathbf{Q}}_a \mathbf{S}_a \right) \hat{\boldsymbol{q}} \\ & = \hat{\boldsymbol{q}}^\top \mathbf{Q} \hat{\boldsymbol{q}} + 2 \boldsymbol{p}^\top \hat{\boldsymbol{q}}, \end{aligned} \tag{6}$$

where $\mathbf{Q}$ and $\mathbf{p}$ are the appropriate matrix and vector with $\hat{\mathbf{Q}}_d = \text{blkdiag}(\mathbf{Q}_d, \cdots, \mathbf{Q}_d)$ and $\hat{\mathbf{Q}}_a = \text{blkdiag}(\mathbf{Q}_a, \cdots, \mathbf{Q}_a)$. Now the cost function is expressed in a quadratic form. Using the above cost function, we reformulate the MPC as follows:

Discrete-Kinematic-MPC:

$$\begin{aligned} \min_{\hat{\boldsymbol{q}}} \quad & \hat{\boldsymbol{q}}^\top \mathbf{Q} \hat{\boldsymbol{q}} + 2 \boldsymbol{p}^\top \hat{\boldsymbol{q}} \\ \text{s.t.} \quad & \hat{\dot{\boldsymbol{q}}}_{\min} \le \mathbf{S}_v \hat{\boldsymbol{q}} + \boldsymbol{v} \le \hat{\dot{\boldsymbol{q}}}_{\max}, \\ & \hat{\boldsymbol{q}}_{\min} \le \hat{\boldsymbol{q}} \le \hat{\boldsymbol{q}}_{\max}, \end{aligned} \tag{7}$$

where $(.)_{\min}$ and $(.)_{\max}$ denote the minimum and maximum values for the bound of $(.)$. The optimization problem in (7) is solvable via QP, allowing us to rapidly compute the joint position command. We will analyze the detailed computation time in terms of the prediction horizons in Section IV.

To ensure stability of the formulated MPC, we enforce additional constraints for the terminal joint position and velocity. If the final prediction of (4) includes the terminal time step ($i + n_p = N$), we need to insert additional inequality constraints $\boldsymbol{q}_N^n - \boldsymbol{\epsilon}_q \le \boldsymbol{q}_N \le \boldsymbol{q}_N^n + \boldsymbol{\epsilon}_q$ and $\boldsymbol{q}_N^n - \boldsymbol{\epsilon}_v \le \boldsymbol{q}_N \le \boldsymbol{q}_N^n + \boldsymbol{\epsilon}_v$ where $\boldsymbol{\epsilon}_q$ and $\boldsymbol{\epsilon}_v$ are allowable small boundaries for the stability verification. These constraints guarantee that the system controlled by the MPC is stable, assuming that the nominal state at the terminal time step is quasi-static or at equilibrium.

C. Nominal Trajectory via Inverse Kinematics

The linearization of $g(\boldsymbol{q})$ relies on the nominal trajectory $(\boldsymbol{\vartheta}_0^d, \cdots, \boldsymbol{\vartheta}_N^d)$, which can be obtained through inverse kinematics. One simple method to solve the inverse kinematics problem is to employ the pseudoinverse of the Jacobian [18]. In the k-th time step, the joint velocity command for executing n_t tasks is computed as

$$\dot{\boldsymbol{q}}_k^{(j)} = \dot{\boldsymbol{q}}_k^{(j-1)} + \left(\mathbf{J}_k^{(j|\text{pre})} \right)^\dagger \left(k^{(j)} \boldsymbol{e}_k^{(j)} - \mathbf{J}_k^{(j)} \dot{\boldsymbol{q}}_k^{(j-1)} \right), \tag{8}$$

where $\dot{\boldsymbol{q}}_k^{(0)} = \mathbf{0}$, $\mathbf{J}_k^{(j|\text{pre})} = \mathbf{J}_k^{(j)} \mathbf{N}_k^{(j-1)}$ and $\mathbf{N}_k^{(j)} = \mathbf{N}_k^{(j-1)} - \left(\mathbf{J}_k^{(j|\text{pre})} \right)^\dagger \mathbf{J}_k^{(j|\text{pre})}$. In addition, the superscript (j) denotes the properties for the j-th task. For instance, $\boldsymbol{e}^{(j)}$ and $k^{(j)} > 0$ are the task error and constant gain for the j-th task, respectively. By recursively computing the above equation (8), the command velocity for the hierarchical tasks is $\dot{\boldsymbol{q}}_k^{(\text{cmd})} = \dot{\boldsymbol{q}}_k^{(n_t)}$. When computing the pseudoinverse of $\mathbf{J}$, we incorporate the compact Singular Value Decomposition (SVD) to prevent the system from diverging as follows:

$$\mathbf{J} = \mathbf{U} \boldsymbol{\Sigma} \mathbf{V}^\top \approx \mathbf{U}_r \boldsymbol{\Sigma}_r \mathbf{V}_r^\top, \tag{9}$$

where $\boldsymbol{\Sigma}_r$ represents the reduced singular value matrix by removing the smaller singular values than the pre-defined threshold. $\mathbf{U}_r$ and $\mathbf{V}_r$ are corresponding to the reduced $\boldsymbol{\Sigma}_r$. Using the compact SVD, we compute the pseudoinverse of the Jacobian as

$$\mathbf{J}^\dagger = \mathbf{V}_r \boldsymbol{\Sigma}_r^{-1} \mathbf{U}_r^\top. \tag{10}$$

Using the above pseudoinverse of the Jacobian, we prevent the robotic system from becoming unstable near or at singular configurations. However, the joint position may abruptly change due to the heuristic threshold for the compact SVD. Our MPC approach is able to resolve this discontinuity issue via additive damping and acceleration terms in the cost function. In Section IV, we will compare the results of the simple inverse kinematics method with our MPC approach.

III. Dynamic MPC with Constraints

A. Nonlinear and Continuous-time Dynamic Model

The rigid body dynamics equation of a manipulator is expressed as follows:

$$\mathbf{D}(\boldsymbol{q}) \ddot{\boldsymbol{q}} + \mathbf{H}(\dot{\boldsymbol{q}}, \boldsymbol{q}) = \boldsymbol{u}, \tag{11}$$

where $\boldsymbol{u} \in \mathbb{R}^n$, $\mathbf{D}(\boldsymbol{q}) \in \mathbb{R}^{n \times n}$, $\mathbf{D}(\boldsymbol{q}) > 0$, and $\mathbf{H}(\dot{\boldsymbol{q}}, \boldsymbol{q}) \in \mathbb{R}^n$ denote the torque input, the mass/inertia matrix, and sum of Coriolis/centrifugal and gravitational forces, respectively. The forward and inverse dynamic equations then

$$\begin{aligned} \text{FD}(\boldsymbol{q}, \dot{\boldsymbol{q}}, \boldsymbol{u}) &= \mathbf{D}(\boldsymbol{q})^{-1} (\boldsymbol{u} - \mathbf{H}(\dot{\boldsymbol{q}}, \boldsymbol{q})) = \ddot{\boldsymbol{q}}, \\ \text{ID}(\boldsymbol{q}, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}}) &= \mathbf{D}(\boldsymbol{q}) \ddot{\boldsymbol{q}} + \mathbf{H}(\dot{\boldsymbol{q}}, \boldsymbol{q})) = \boldsymbol{u}. \end{aligned} \tag{12}$$

In later discussions, we use simplified notations $\mathbf{D}$, $\mathbf{H}$, $\mathbf{J}$, and $\dot{\mathbf{J}}$ for $\mathbf{D}(\boldsymbol{q})$, $\mathbf{H}(\dot{\boldsymbol{q}}, \boldsymbol{q})$, $\mathbf{J}(\boldsymbol{q})$, and $\dot{\mathbf{J}}(\dot{\boldsymbol{q}}, \boldsymbol{q})$.

Now, defining a state $\boldsymbol{x} = [\boldsymbol{q}^\top, \dot{\boldsymbol{q}}^\top]^\top \in \mathbb{R}^{2n}$ a continuous-time state space model can be expressed as follows:

$$\dot{\boldsymbol{x}} = f(\boldsymbol{x}, \boldsymbol{u}) = \begin{bmatrix} \dot{\boldsymbol{q}} \\ \text{FD}(\boldsymbol{q}, \dot{\boldsymbol{q}}, \boldsymbol{u}) \end{bmatrix}, \tag{13}$$

where $f : \mathbb{R}^{2n} \times \mathcal{U} \to \mathbb{R}^{2n}$ and is nonlinear, and $\mathcal{U} \subseteq \mathbb{R}^n$. Since we want to formulate a QP problem, we need to linearize and discretize the above state-space model to formulate the MPC in the shape of the QP problem.

B. Discrete and Linear State-Space Model

We consider the finite-time horizon $\mathcal{T} = [t_i, t_f]$ and normalize the time domain by using a dilation coefficient $\sigma = t_i - t_f$. The normalized variable is defined as $\tau = \sigma^{-1}(t - t_i) \in [0, 1]$ for the unit interval. Thus we have

$$\dot{\boldsymbol{x}}_\tau = \frac{d\boldsymbol{x}_\tau}{dt} = \frac{1}{\sigma}\frac{d\boldsymbol{x}_\tau}{d\tau} = f(\boldsymbol{x}_\tau, \boldsymbol{u}_\tau). \tag{14}$$

The dynamics model in the normalized time domain is linearized along a given nominal trajectory $(\boldsymbol{x}_\tau^n, \boldsymbol{u}_\tau^n)$:

$$d\boldsymbol{x}_\tau \approx (\mathbf{A}_\tau \boldsymbol{x}_\tau + \mathbf{B}_\tau \boldsymbol{u}_\tau + \boldsymbol{r}_\tau)d\tau, \tag{15}$$

where $\mathbf{A}_\tau = \sigma \nabla_{\boldsymbol{x}} f(\boldsymbol{x}, \boldsymbol{u})|_{(\boldsymbol{x}_\tau^n, \boldsymbol{u}_\tau^n)}$, $\mathbf{B}_\tau = \sigma \nabla_{\boldsymbol{u}} f(\boldsymbol{x}, \boldsymbol{u})|_{(\boldsymbol{x}_\tau^n, \boldsymbol{u}_\tau^n)}$, and $\boldsymbol{r}_\tau = \sigma f(\boldsymbol{x}_\tau^n, \boldsymbol{u}_\tau^n) - \mathbf{A}_\tau \boldsymbol{x}_\tau^n - \mathbf{B}_\tau \boldsymbol{u}_\tau^n$. In the above formulations of $\mathbf{A}_\tau$ and $\mathbf{B}_\tau$, we utilize the partial derivative of Lagrangian expressions of forward dynamics which are

$$\begin{aligned} \frac{\partial \mathrm{FD}}{\partial \boldsymbol{q}} &= \frac{\partial \mathbf{D}^{-1}}{\partial \boldsymbol{q}}(\boldsymbol{u} - \mathbf{H}) - \mathbf{D}^{-1}\frac{\partial \mathbf{H}}{\partial \boldsymbol{q}}, \\ \frac{\partial \mathrm{FD}}{\partial \dot{\boldsymbol{q}}} &= -\mathbf{D}^{-1}\frac{\partial \mathbf{H}}{\partial \dot{\boldsymbol{q}}}, \quad \frac{\partial \mathrm{FD}}{\partial \boldsymbol{u}} = \mathbf{D}^{-1}, \end{aligned} \tag{16}$$

and $\frac{\partial \mathbf{D}^{-1}}{\partial \boldsymbol{q}} = -\mathbf{D}^{-1}\frac{\partial \mathbf{D}}{\partial \boldsymbol{q}}\mathbf{D}^{-1}$. From [5], the relationship between the derivatives of inverse and forward dynamics are

$$\left.\frac{\partial \mathrm{FD}}{\partial \boldsymbol{\xi}}\right|_{(\boldsymbol{q}_\tau^n, \dot{\boldsymbol{q}}_\tau^n, \boldsymbol{u}_\tau^n)} = -\mathbf{D}(\boldsymbol{q}_\tau^n)^{-1}\left.\frac{\partial \mathrm{ID}}{\partial \boldsymbol{\xi}}\right|_{(\boldsymbol{q}_\tau^n, \dot{\boldsymbol{q}}_\tau^n, \ddot{\boldsymbol{q}}_\tau^n)}, \tag{17}$$

where $\boldsymbol{\xi} \in \{\boldsymbol{q}, \dot{\boldsymbol{q}}\}$. $\frac{\partial \mathrm{ID}}{\partial \boldsymbol{q}}$ and $\frac{\partial \mathrm{ID}}{\partial \dot{\boldsymbol{q}}}$ are directly obtained by the recursive Newton-Euler algorithm.[1] In this study, we employ the computation algorithm proposed in [5] to obtain the partial derivative terms.

We convert the continuous-time state space model to discrete time by integrating the above differential equation:

$$\int_{\tau_k}^{\tau_k + \Delta\tau} d\boldsymbol{x}_\tau = \int_{\tau_k}^{\tau_k + \Delta\tau} (\mathbf{A}_\tau \boldsymbol{x}_\tau + \mathbf{B}_\tau \boldsymbol{u}_\tau + \boldsymbol{r}_\tau)d\tau, \tag{18}$$

where we set $\Delta\tau = \Delta t$. Then, the discrete-time state space model is obtained as follows:

$$\boldsymbol{x}_{k+1} = \mathbf{A}_k \boldsymbol{x}_k + \mathbf{B}_k \boldsymbol{u}_k + \boldsymbol{r}_k, \tag{19}$$

where $\mathbf{A}_k = \mathbf{A}_{\tau_k}\Delta t + \mathbf{I}$, $\mathbf{B}_k = \mathbf{B}_{\tau_k}\Delta t$, and $\boldsymbol{r}_k = \boldsymbol{r}_{\tau_k}\Delta t$ with $k \in \{i, \cdots, i + n_p\}$. Considering the concatenated state and control input vectors: $\hat{\boldsymbol{x}}_i = [\boldsymbol{x}_i^\top, \cdots, \boldsymbol{x}_{i+n_p}^\top]^\top$, $\hat{\boldsymbol{u}}_i = [\boldsymbol{u}_i^\top, \cdots, \boldsymbol{u}_{i+n_p-1}^\top]^\top$, and $\hat{\boldsymbol{r}}_i = [\boldsymbol{r}_i^\top, \cdots, \boldsymbol{r}_{i+n_p-1}^\top]^\top$, we formulate the discrete-time state model in a similar form to [17] as

$$\hat{\boldsymbol{x}}_i = \hat{\mathbf{A}}_i \boldsymbol{x}_i + \hat{\mathbf{B}}_i \hat{\boldsymbol{u}}_i + \mathbf{G}_i \hat{\boldsymbol{r}}_i, \tag{20}$$

where $\hat{\mathbf{A}}_i = \boldsymbol{\Omega}(i)$,

$$\begin{aligned} \hat{\mathbf{B}}_i &= [\boldsymbol{\Omega}(i+1)\mathbf{B}_i, \cdots, \boldsymbol{\Omega}(i+n_p)\mathbf{B}_{i+n_p-1}], \\ \mathbf{G}_i &= [\boldsymbol{\Omega}(i+1), \cdots, \boldsymbol{\Omega}(i+n_p)], \\ \boldsymbol{\Omega}(s) &= [\boldsymbol{\Phi}(i,s)^\top, \cdots, \boldsymbol{\Phi}(i+n_p, s)^\top]^\top. \end{aligned} \tag{21}$$

[1]Implementation of these rigid-body dynamics and partial derivative computations is available in the open-source Pinocchio library: https://github.com/stack-of-tasks/pinocchio

In addition, the matrix $\boldsymbol{\Phi}(j, s)$ is computed as follows:

$$\boldsymbol{\Phi}(j,s) = \begin{cases} \mathbf{A}_{j-1} \cdots \mathbf{A}_s & \text{when } j \geq s+1 \\ \mathbf{I} & \text{when } j = s \\ \mathbf{0} & \text{otherwise.} \end{cases} \tag{22}$$

The above linear state-space model in (20) is rearranged in terms of a decision variable $\boldsymbol{z}_i = [\hat{\boldsymbol{x}}_i^\top, \hat{\boldsymbol{u}}_i^\top]^\top$ as

$$\begin{bmatrix} \mathbf{I} & -\hat{\mathbf{B}}_i \end{bmatrix} \mathbf{z}_i = \hat{\mathbf{A}}_i \boldsymbol{x}_i + \mathbf{G}_i \hat{\boldsymbol{r}}_i. \tag{23}$$

Now, we consider the constraint corresponding to the nonlinear dynamics of robots as a linear constraint in terms of our decision variable $\boldsymbol{z}_i$.

C. MPC formulation

Similar to the kinematic MPC formulation, we formulate dynamic MPC, including torque limit constraints. With prediction horizon n_p, the optimization problem is defined as follows:

$$\begin{aligned} \min_{\hat{\boldsymbol{x}}_i, \hat{\boldsymbol{u}}_i} \quad & \mathcal{J} = \sum_{k=i}^{i+n_p} \Big((\boldsymbol{\vartheta}_k^d - g(\boldsymbol{q}_k))^\top \mathbf{Q}_e (\boldsymbol{\vartheta}_k^d - g(\boldsymbol{q}_k)) \\ & \qquad + \dot{\boldsymbol{q}}_k^\top \mathbf{Q}_d \dot{\boldsymbol{q}}_k + \boldsymbol{u}_k^\top \mathbf{Q}_u \boldsymbol{u}_k \Big), \\ \text{s.t.} \quad & \boldsymbol{x}_{k+1} = \mathbf{A}_k \boldsymbol{x}_k + \mathbf{B}_k \boldsymbol{u}_k + \boldsymbol{r}_k, \\ & \boldsymbol{u}_{\min} \leq \boldsymbol{u}_k \leq \boldsymbol{u}_{\max}, \quad \forall k \in \{i, \cdots, i + n_p - 1\}, \\ & \dot{\boldsymbol{q}}_{\min} \leq \dot{\boldsymbol{q}}_j \leq \dot{\boldsymbol{q}}_{\max}, \\ & \boldsymbol{q}_{\min} \leq \boldsymbol{q}_j \leq \boldsymbol{q}_{\max}, \quad \forall j \in \{i, \cdots, i + n_p\}, \end{aligned} \tag{24}$$

where $\mathbf{Q}_u$ denotes the weighting matrix for the torque input term. We approximate the tracking error term in the cost function by using (5):

$$\begin{aligned} \mathcal{J} &\approx \left(\sum_{k=i}^{i+n_p} \boldsymbol{q}_k^\top \mathbf{J}_k^{n\top} \mathbf{Q}_e \mathbf{J}_k^n \mathbf{q}_k - 2\boldsymbol{q}_k^{n\top} \mathbf{J}_k^n \mathbf{Q}_e \boldsymbol{q}_k \right) \\ &\quad + \hat{\dot{\boldsymbol{q}}}_i^\top \hat{\mathbf{Q}}_d \hat{\dot{\boldsymbol{q}}}_i + \hat{\boldsymbol{u}}_i^\top \hat{\mathbf{Q}}_u \hat{\boldsymbol{u}}_i \\ &= \boldsymbol{z}_i^\top \mathbf{W}_i \boldsymbol{z}_i + 2\boldsymbol{w}_i^\top \boldsymbol{z}_i, \end{aligned} \tag{25}$$

where $\hat{\mathbf{Q}}_u = \mathrm{blkdiag}(\mathbf{Q}_u, \cdots, \mathbf{Q}_u)$. In addition, $\mathbf{W}_i$ and $\boldsymbol{w}_i$ are the proper matrix and vector for the quadratic cost function in terms of the decision variable $\boldsymbol{z}_i$. Finally, we reformulate the optimization problem in terms of the decision variable $\boldsymbol{z}_i$ as follows:

Discrete-Dynamic-MPC:

$$\begin{aligned} \min_{\boldsymbol{z}_i} \quad & \boldsymbol{z}_i^\top \mathbf{W}_i \boldsymbol{z}_i + 2\boldsymbol{w}_i^\top \boldsymbol{z}_i \\ \text{s.t.} \quad & \begin{bmatrix} \mathbf{I} & -\hat{\mathbf{B}}_i \end{bmatrix} \boldsymbol{z}_i = \hat{\mathbf{A}}_i \boldsymbol{x}_i + \mathbf{G}_i \hat{\boldsymbol{r}}_i, \\ & \boldsymbol{z}_{\min} \leq \boldsymbol{z}_i \leq \boldsymbol{z}_{\max}, \end{aligned} \tag{26}$$

where $\boldsymbol{z}_{\min}$ and $\boldsymbol{z}_{\max}$ are the minimum and maximum bounds for the decision variable $\boldsymbol{z}_i$. Although the dimension of the decision variable in the dynamic MPC ($\dim(\boldsymbol{z}_i) = 3n_p n - n$) is larger than that of kinematic MPC($\dim(\boldsymbol{q}) = n_p n$), the reformulated optimization problem is a QP, which is much faster to solve than nonlinear MPC. Similar to the kinematic MPC, we enforce the additional state constraints for the stability verification if the last prediction step is at the end of the trajectory.

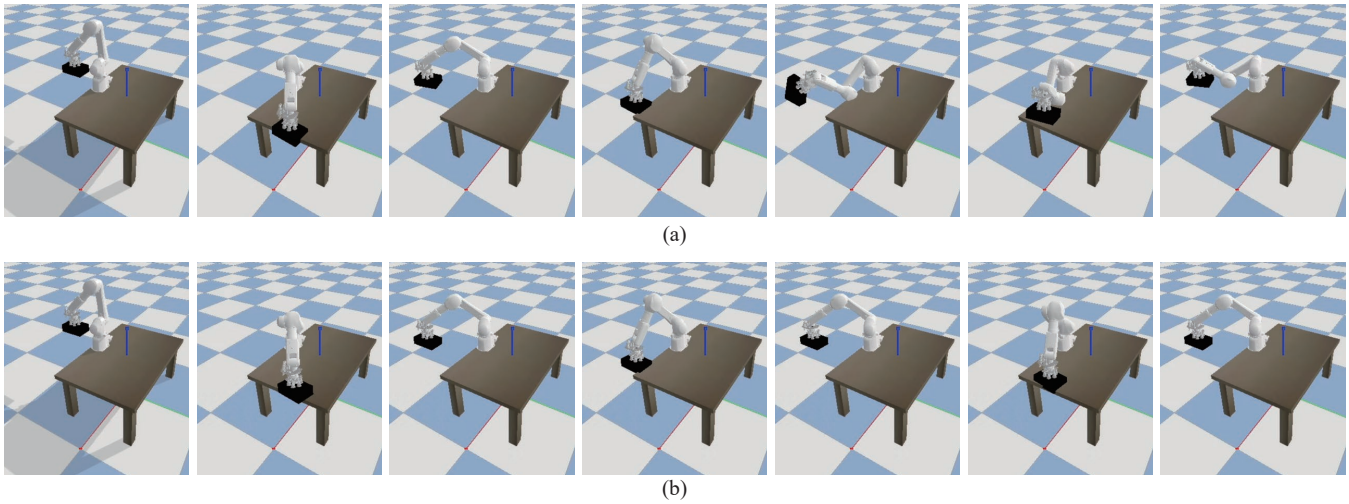


Fig. 2: **Simulation snapshots**: (a) Task trajectory tracking using OSC, (b) Task trajectory tracking using the proposed MPC

D. Nominal Trajectory via Inverse Dynamics

The proposed MPC relies on the linearized state-space model of robot dynamics. For this reason, it is important to generate a realistic nominal trajectory, which can be done using an inverse dynamics controller, for example via OSC [28]. When computing a dynamically consistent inverse of the Jacobian, we also utilize the compact SVD to prevent the robot from becoming unstable near singular configurations.

IV. Simulations and Experiments

In this section, we validate the proposed QP-based MPC approach using two Kawasaki manipulators: RS007N (simulation) and RS020N (experiment). In the simulation, the robot is controlled by torque commands, while the real robot is controlled by a joint position controller provided by Kawasaki control boxes. We validate the effectiveness of the devised dynamic MPC by controlling the manipulator (RS007N) with a heavy payload in the PyBullet simulation environment [6]. We use *QuadProg++*[2], which is based on the Goldfarb-Idinani active-set dual method, to solve the formulated QP problems on a laptop with a i7-8650U CPU and 16 GB of RAM. In addition, the experimental work is demonstrated using the real robot (an RS020N with supporting software provided by Dexterity).[3]

A. Handling a payload with torque limits

In this simulation, we aim to track a desired end-effector's trajectory used in Dexterity applications while picking and placing parcels. The weight of the parcel is 12 kg, which is heavier than those of normal packages. We consider the joint torque limits as $[239,\ 239,\ 124.5,\ 32,\ 40.96,\ 25.6]\ Nm$. In addition, it is assumed that the suction gripper's capacity is enough to handle the payload. For OSC, the end-effector position task is higher than the end-effector orientation task. We set the PD gains empirically for the above tasks as $\mathbf{K}^p_{\text{pos}} = [100,\ 100,\ 100]$, $\mathbf{K}^d_{\text{pos}} = [7,\ 13,\ 7]$, $\mathbf{K}^p_{\text{ori}} = [20,\ 20,\ 20]$, and $\mathbf{K}^d_{\text{ori}} = [1.5,\ 1.5,\ 1.5]$. We compute the orientation errors by using quaternion product then convert them to 3-dimensional vectors. In addition, the gains of the joint impedance controller are $\mathbf{K}^p_{\text{imp}} = [100,\ 100,\ 100,\ 50,\ 50,\ 1]$ and $\mathbf{K}^d_{\text{imp}} = [3,\ 5,\ 5,\ 0.2,\ 0.2,\ 0.1]$. We use two diagonal weighting matrices $\mathbf{Q}_e = \text{diag}(10, \cdots, 10)$ and $\mathbf{Q}_d = \text{diag}(0.0001, \cdots, 0.0001)$, without $\mathbf{Q}_u$.

[2]QuadProg++: `https://github.com/liuq/QuadProgpp`
[3]Dexterity, Inc.: `https://www.dexterity.ai`

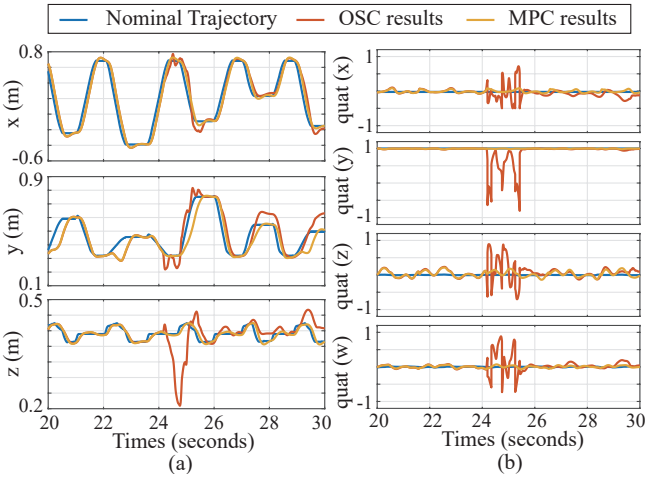


Fig. 3: **Pose of the end-effector**: (a) the position of the end-effector, (b) the orientation of the end-effector.

Fig. 2 (a) and (b) show the snapshots of the simulations demonstrating OSC and MPC with the same task trajectory, respectively. As shown in Fig. 2 (a), the robot controlled by OSC abruptly changes the configuration due to the heavy payload (see the 5th snapshot in Fig. 2 (a)). On the other hand, the proposed MPC prevents the rapid change of the configuration while tracking the given task trajectory. The above behavioral difference is shown in Fig. 3, including the desired task trajectory (blue line) and the results controlled by OSC (orange line) and MPC (yellow line). More specifically, the end-effector's position in the z direction and its orientation significantly fluctuate between 24 seconds to 26 seconds. Although the MPC results are not perfectly tracking the trajectory due to the payload, the overall tracking error of MPC is much smaller than OSC.

The detailed comparison of the tracking error is based on the accumulated tracking error norm defined as $\text{err} = \sum_{k=0}^{n_a} \|\boldsymbol{e}_k^{(\text{pos, ori})}\|_2$ where n_a denotes the time horizon for accumulating the norm of task errors. The accumulated errors

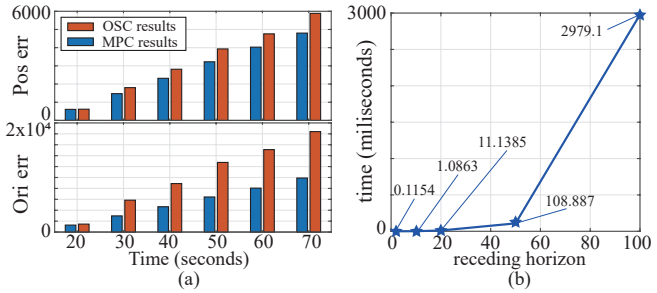


Fig. 4: (a) accumulated errors of the tasks, (b) the computation time in terms of the receding horizon

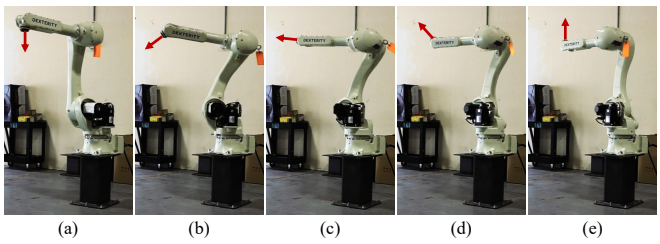


Fig. 5: **Experiment snapshots**: The red arrows visualize the orientation of the end-effector. (a) initial configuration, (b) moving toward singular configuration, (c) moving through singularity, (d) moving after pass through singularity, (e) final configuration

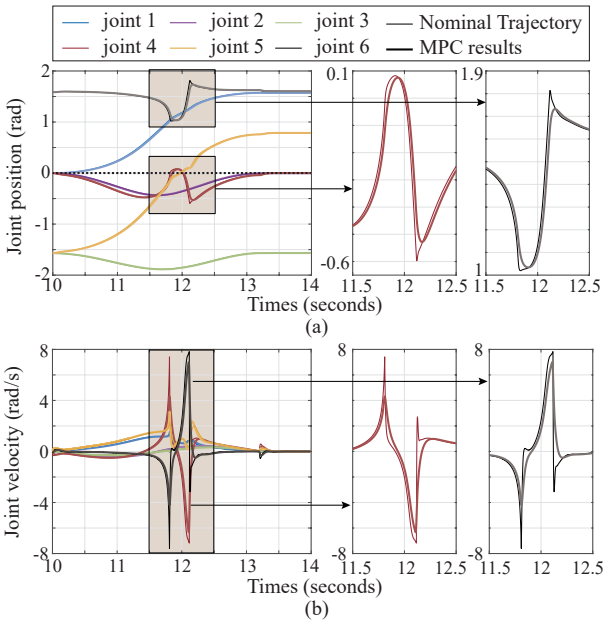


Fig. 6: **Experiment results**: (a) joint position, (b) joint velocity. The plots around the singularity are magnified in the right-sided plots.

of two main tasks are described in Fig 4 (a). Overall, the tracking errors of both tasks controlled by the proposed MPC are clearly smaller than those controlled by OSC for all times. As shown in Fig. 4 (a), the gap between the orientation errors increases significantly since the orientation task is lower prioritized in the hierarchical control. In addition, we analyze the computation time in terms of the receding horizon as presented in Fig. 4 (b). The proposed MPC approach is formulated and solved as a QP, so the computation time depends on the dimension of the decision variables (complexity is $O(n_p^3)$ for number of receding horizon timesteps n_p). A receding horizon of 10 milliseconds with $n_p = 10$ can be implemented with 1 kHz closed-loop controller, even given the limited specs of the laptop.

B. Singularity-free manipulation

We verify our MPC by demonstrating a real experiment using a RS020N robot. The initial and final configurations are $[0,\ 0,\ -\frac{\pi}{2},\ 0,\ -\frac{\pi}{2},\ \frac{\pi}{2}]$ rad and $[\frac{\pi}{2},\ 0,\ -\frac{\pi}{2},\ 0,\ \frac{\pi}{4},\ \frac{\pi}{2}]$ rad, respectively. The end-effector's position and orientation trajectories are generated using cubic spline and quaternion interpolation with 4-second time durations. To perform the task trajectories, the robot must pass through a singularity when joint 5 is at zero. The weighting matrices for MPC are $\mathbf{Q}_e = \text{diag}(2000, \cdots, 2000)$ and $\mathbf{Q}_d = \text{diag}(0.01, \cdots, 0.01)$, without $\mathbf{Q}_u$. Since the embedded joint space controller controls the real robot, we execute the dynamic simulation using the torque command and then generate the joint trajectory based on the measured joint position from the simulation.

Fig. 5 represents the snapshots of the experiment using the proposed MPC. While moving from the initial configuration (Fig. 5 (a)) to the final one (Fig. 5 (e)), the robot avoids the singularity by twisting the last three joints (Fig. 5 (c)). On the other hand, the robot terminates the operation near the singularity when using a classical OSC controller. The measured joint position and velocity are shown in Fig. 6. The regular and bold lines in Fig. 6 are the nominal trajectories computed by OSC and the results controlled by the proposed MPC, respectively. The dotted line in Fig. 6 represents the location of the singularity, specifically where q_5 equals zero. As q_5 approaches zero, the joints q_4 and q_6 experience sudden changes to follow the trajectory using the classical OSC with the compact SVD. However, our proposed MPC approach offers the advantage of smoothing the joint position and velocity near the singularity, as depicted in Fig. 6. In particular, we observe significant smoothness in the velocity level compared with the nominal trajectory as shown in Fig 6 (b). These experimental results show that the proposed MPC effectively avoids singularity while executing multiple tasks.

V. Conclusion

This paper proposes a real-time model predictive control (MPC) framework that efficiently executes hierarchical tasks while ensuring safe movement through singularity with low-level feedback controllers that take into account both kinematics and dynamics. Our proposed approach consists of first generating nominal trajectories using hierarchical control, linearizing along the nominal trajectories, and optimizing via a QP-based MPC formulation. We analyze the computation time of our framework for the underlying receding horizon and achieve real-time MPC rates of 1 kHz for feedback control. The simulation and experiment results show that our proposed optimization framework successfully handles heavy payload tasks while meeting torque limits and safely passes through singularities to produce smooth motions.

Moving forward, we plan to employ the proposed MPC framework to more complicated and realistic scenarios for logistics. Furthermore, we are interested in resolving stability issues which arises when the actual state is deviated far from the nominal points. In addition, we will consider learning the linearized model more accurately to reduce the tracking error in cases with unknown and time-varying payloads.

REFERENCES

[1] F. Aghili, "Impedance control of manipulators carrying a heavy payload," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2009, pp. 3410–3415.

[2] L. Baccelliere *et al.*, "Development of a human size and strength compliant bi-manual platform for realistic heavy manipulation tasks," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2017, pp. 5594–5601.

[3] M. Bednarczyk, H. Omran, and B. Bayle, "Model predictive impedance control," in *Proceedings of the IEEE International Conference on Robotics and Automation*, 2020, pp. 4702–4708.

[4] D. J. Braun, Y. Chen, and L. Li, "Operational space control under actuation constraints using strictly convex optimization," *IEEE Transactions on Robotics*, vol. 36, no. 1, pp. 302–309, 2019.

[5] J. Carpentier and N. Mansard, "Analytical derivatives of rigid body dynamics algorithms," in *Robotics: Science and systems*, 2018.

[6] E. Coumans and Y. Bai, *PyBullet, a Python module for physics simulation for games, robotics and machine learning*, `http://pybullet.org`, 2016–2021.

[7] A. Dietrich, C. Ott, and J. Park, "The hierarchical operational space formulation: Stability analysis for the regulation case," *IEEE Robotics and Automation Letters*, vol. 3, no. 2, pp. 1120–1127, 2018.

[8] W. Echelmeyer, A. Kirchheim, and E. Wellbrock, "Robotics-logistics: Challenges for automation of logistic processes," in *Proceedings of the IEEE International Conference on Automation and Logistics*, 2008, pp. 2099–2103.

[9] R. Grandia, F. Farshidian, R. Ranftl, and M. Hutter, "Feedback MPC for torque-controlled legged robots," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, 2019, pp. 4730–4737.

[10] G. P. Incremona, A. Ferrara, and L. Magni, "MPC for robot manipulators with integral sliding modes generation," *IEEE/ASME Transactions on Mechatronics*, vol. 22, no. 3, pp. 1299–1307, 2017.

[11] E. Kang, H. Qiao, Z. Chen, and J. Gao, "Tracking of uncertain robotic manipulators using event-triggered model predictive control with learning terminal cost," *IEEE Transactions on Automation Science and Engineering*, 2022.

[12] Z.-H. Kang, C.-A. Cheng, and H.-P. Huang, "A singularity handling algorithm based on operational space control for six-degree-of-freedom anthropomorphic manipulators," *International Journal of Advanced Robotic Systems*, vol. 16, no. 3, p. 1 729 881 419 858 910, 2019.

[13] O. Khatib, "A unified approach for motion and force control of robot manipulators: The operational space formulation," *IEEE Journal on Robotics and Automation*, vol. 3, no. 1, pp. 43–53, 1987.

[14] S. Kleff, A. Meduri, R. Budhiraja, N. Mansard, and L. Righetti, "High-frequency nonlinear model predictive control of a manipulator," in *Proceedings of the IEEE International Conference on Robotics and Automation*, 2021, pp. 7330–7336.

[15] J. Koenemann *et al.*, "Whole-body model-predictive control applied to the HRP-2 humanoid," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2015, pp. 3346–3351.

[16] D. Lee, W. Lee, J. Park, and W. K. Chung, "Task space control of articulated robot near kinematic singularity: Forward dynamics approach," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 752–759, 2020.

[17] J. Lee, S. H. Bang, E. Bakolas, and L. Sentis, "MPC-based hierarchical task space control of underactuated and constrained robots for execution of multiple tasks," in *Proceedings of the IEEE Conference on Decision and Control*, 2020, pp. 5942–5949.

[18] J. Lee, N. Mansard, and J. Park, "Intermediate desired value approach for task transition of robots in kinematic control," *IEEE Transactions on Robotics*, vol. 28, no. 6, pp. 1260–1277, 2012.

[19] G. J. Maeda, S. P. Singh, and D. C. Rye, "Improving operational space control of heavy manipulators via open-loop compensation," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2011, pp. 725–731.

[20] D. Q. Mayne, J. B. Rawlings, C. V. Rao, and P. O. Scokaert, "Constrained model predictive control: Stability and optimality," *Automatica*, vol. 36, no. 6, pp. 789–814, 2000.

[21] M. Mistry and L. Righetti, "Operational space control of constrained and underactuated systems," in *Robotics: Science and systems*, vol. 7, 2012, pp. 225–232.

[22] M. A. Murtaza, S. Aguilera, V. Azimi, and S. Hutchinson, "Real-time safety and control of robotic manipulators with torque saturation in operational space," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, IEEE, 2021, pp. 702–708.

[23] J. Nakanishi, R. Cory, M. Mistry, J. Peters, and S. Schaal, "Operational space control: A theoretical and empirical comparison," *International Journal of Robotics Research*, vol. 27, no. 6, pp. 737–757, 2008.

[24] D. Nicolis, F. Allevi, and P. Rocco, "Operational space model predictive sliding mode control for redundant manipulators," *IEEE Transactions on Robotics*, vol. 36, no. 4, pp. 1348–1355, 2020.

[25] J. Nubert, J. Köhler, V. Berenz, F. Allgöwer, and S. Trimpe, "Safe and fast tracking on a robot manipulator: Robust MPC and neural network control," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 3050–3057, 2020.

[26] H. Sadeghian, L. Villani, M. Keshmiri, and B. Siciliano, "Task-space control of robot manipulators with null-space compliance," *IEEE Transactions on Robotics*, vol. 30, no. 2, pp. 493–506, 2013.

[27] A. S. Sathya, W. Decre, G. Pipeleers, and J. Swevers, "A simple formulation for fast prioritized optimal control of robots using weighted exact penalty functions," in *Proceedings of the IEEE International Conference on Robotics and Automation*, IEEE, 2022, pp. 5262–5269.

[28] L. Sentis and O. Khatib, "Synthesis of whole-body behaviors through hierarchical control of behavioral primitives," *International Journal of Humanoid Robotics*, vol. 2, no. 04, pp. 505–518, 2005.