Hierarchical Task Model Predictive Control for Sequential Mobile Manipulation Tasks

Xintong Du, Siqi Zhou, *Member, IEEE*, and Angela P. Schoellig

***Abstract*—Mobile manipulators are envisioned to serve more complex roles in people's everyday lives. With recent breakthroughs in large language models, task planners have become better at translating human verbal instructions into a sequence of tasks. However, there is still a need for a decision-making algorithm that can seamlessly interface with the high-level task planner to carry out the sequence of tasks efficiently. In this work, building on the idea of nonlinear lexicographic optimization, we propose a novel Hierarchical-Task Model Predictive Control framework that is able to complete sequential tasks with improved performance and reactivity by effectively leveraging the robot's redundancy. Compared to the state-of-the-art task-prioritized inverse kinematic control method, our approach has improved hierarchical trajectory tracking performance by 42% on average when facing task changes, robot singularity, and reference variations. Compared to a typical single-task architecture, our proposed hierarchical task control architecture enables the robot to traverse a shorter path in task space and achieves an execution time 2.3 times faster when executing a sequence of delivery tasks. We demonstrated the results with real-world experiments on a 9 degrees of freedom mobile manipulator.**

***Index Terms*—Mobile manipulation, redundant robots, whole-body motion planning and control.**

I. Introduction

Taking the form of a mobile base with robotic arms mounted on top, mobile manipulators have always been envisioned to serve more complex service roles in people's everyday lives. These roles require mobile manipulators to perform a sequence of manipulation tasks scattered at distant locations in a complex environment to fulfill high-level tasks instructed by a human [1]. We refer to this type of task as the sequential mobile manipulation task (see Fig. 1).

Most control architectures for sequential mobile manipulation tasks consist of cascaded modules running at different frequencies Fig. 2. Given an instruction, task planners decompose it into a sequence of tasks to be executed by the subsequent motion planning and control modules. With recent breakthroughs in large language models, task planners have become better at understanding human verbal instructions. However, the motion planning and control modules could still be improved to enhance optimality and/or reactivity. Typical motion planning and control methods for executing sequential tasks treat each task separately [2], [3], [4]. As a result, robots can only execute one task after another and, for safety concerns, may need to come to a complete stop in between. Efficiency can be significantly improved if robots can attempt multiple tasks by leveraging their kinematic redundancy. For example, mobile manipulators should be able to perform an end effector (EE) task while moving its base towards its next goal. It should also be able to stop moving its base immediately before compromising the EE task. In addition to optimality and efficiency, it is also important to keep robots reactive to task changes or dynamic objects that are often seen in the real world. However, most methods that leverage the robot's redundancy take a planning-oriented approach which puts a strong emphasis on optimality over reactivity [5], [6], [7], [8].

In control literature, leveraging the robot's kinematic redundancy for executing ordered tasks is known as the hierarchical task control problem and has been widely studied in the past decades [9]. In this work, our goal is to solve the sequential mobile manipulation problem using the hierarchical task control framework. We aim to solve the problem with low computational costs to maintain sufficient reactivity to real-world changes or disturbances. Although sequential mobile manipulation can involve a wide variety of tasks, similar to [8], [10], we focus on one typical example where human inputs are interpreted

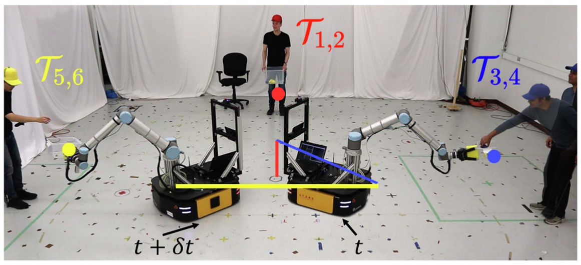

Fig. 1. Our mobile manipulator executes a sequence of delivery tasks while leveraging redundancy for efficiency. Our robot needs to deliver to/pick up from three people following the order, Red, Blue, Yellow. The high-level goal is decomposed into a sequence of alternating base trajectories (strips) and end effector waypoints (dots). At time t_0, the robot leverages its redundancy to the current EE task $\mathcal{T}_4$ (Blue dot) to perform the subsequent base task $\mathcal{T}_5$ (Yellow strip) so that it can reach its next EE waypoint $\mathcal{T}_6$ faster. In this test, our proposed HTMPC motion control architecture is 2.3 times faster than the typical single-task method. A complete video can be found at http://tiny.cc/htmpc.

Manuscript received 12 August 2023; accepted 6 December 2023. Date of publication 13 December 2023; date of current version 28 December 2023. This letter was recommended for publication by Associate Editor A. Del Prete and Editor A. Kheddar upon evaluation of the reviewers' comments. This work was supported by in part by the Natural Sciences and Engineering Research Council of Canada, in part by the Canadian Institute for Advanced Research, and in part by the Alexander Humboldt Foundation. *(Corresponding author: Xintong Du.)*

The authors are with the Learning Systems and Robotics Lab, Technical University of Munich, 80333 München, Germany, also with the Institute for Aerospace Studies, University of Toronto, Toronto, ON M5S 1A1, Canada, and also with the Munich Institute of Robotics and Machine Intelligence (MIRMI), Vector Institute for Artificial Intelligence, University of Toronto Robotics Institute, Toronto, ON M5S 1A4, Canada (e-mail: xintong.du@utias.utoronto.ca; siqi.zhou@robotics.utias.utoronto.ca; angela.schoellig@tum.de).

This letter has supplementary downloadable material available at https://doi.org/10.1109/LRA.2023.3342671, provided by the authors.

Digital Object Identifier 10.1109/LRA.2023.3342671

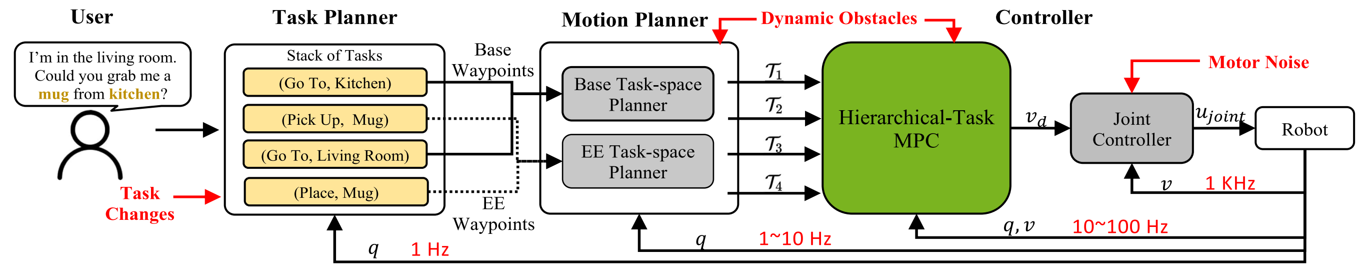

Fig. 2. Proposed HTMPC planning and control architecture for sequential mobile manipulation tasks. Autonomy modules are represented as blocks. Arrows indicate the direction of information flow. Feedback loops from the robot are presented along with their desired close-loop frequency for reactive behaviours. External disturbances or changes coming from the robot or environment are specified as incoming arrows to each autonomy module affected.

as a sequence of alternating base and EE waypoints by the task planner (Fig. 2). However, it is worth mentioning that our proposed method does not assume a specific type of task sequence except that all tasks are tracking tasks.

Our key contributions are as follows:

1) Formulate the sequential mobile manipulation control problem as a Hierarchical Task Model Predictive Control (HTMPC) problem and introduce a reformulation of the lexicographic optimality constraint in HTMPC to make it feasible to solve online.
2) Propose a novel motion planning and control architecture optimized for the HTMPC controller to leverage the robot's kinematic redundancy for sequential tasks.
3) Show that our proposed HTMPC outperforms the state-of-the-art inverse-kinematic-based method when facing task changes, robot singularity, and variations in trajectories. Show that the proposed hierarchical-task control architecture has improved efficiency and reactivity in robot behaviours for sequential tasks compared to existing control architectures.
4) Demonstrate the above results in experiments on a 9 degrees of freedom (DoF) mobile manipulator

Notation: Subscript k denotes the discrete time index over the prediction horizon. We use $\bar{\mathbf{p}}_{i:j}$ to denote the vertically concatenated vector of $(\mathbf{p}_i, \mathbf{p}_{i+1}, \ldots, \mathbf{p}_j)$. The notation for state and control input sequence in MPC is $\bar{\mathbf{x}}_{0:N}$, $\bar{\mathbf{u}}_{0:N-1}$ where N is the prediction window size. We will omit the subscripts in MPC state and control input sequence for simplicity and write them as $\bar{\mathbf{x}}$, $\bar{\mathbf{u}}$.

II. Related Work

A. Sequential Mobile Manipulation Control Architecture

Typical motion control methods for mobile manipulators use a task-space planner to generate trajectories for each EE or base waypoint in the sequence [2], [10]. The EE and base task-space planners work independently; consequently, the resulting plans are not coordinated either in space or time. To avoid conflicting motions, these plans are usually executed sequentially by either decoupled arm and base controllers [2] or whole-body controllers [3], [4]. Regardless of which type of closed-loop controller is used, these methods are single-task approaches in nature as each task-space plan is executed individually. In this work, we aim to improve efficiency significantly by leveraging the robot's redundancy to achieve multiple tasks, i.e. redundancy resolution.

For sequential mobile manipulation, redundancy resolution means coordinating the arm and base toward multiple tasks in the sequence. This can be achieved by using a whole-body planner to generate optimal joint-space trajectories given a sequence of waypoints [5], [6], [7], [8]. However, whole-body planners can be expensive for high DoF robots; it is difficult to re-plan joint-space plans to be reactive in task space. In this work, we re-distributed the workload between the motion planner and controller by interfacing them at the task-space level to achieve reactive robot behaviours in task space. In [10], [11], a set of trajectory design strategies was proposed to generate an arm-base coordinated trajectory to enable "Manipulation on the Move" (MotM). Although kinematic consistency is not guaranteed, this strategy has been proven effective and reliable for sequential pick-and-place tasks. In this work, we took a more general approach, hierarchical-task control, which applies to all trajectory tracking tasks with guaranteed kinematic consistency by directly reasoning with robot kinematics.

B. Hierarchical Task Control

Hierarchical task control leverages robot redundancy to achieve multiple control objectives with different priorities. Optimality for a task with a higher priority should not be compromised by lower-level tasks. As a type of multi-objective optimization problem, hierarchical task control can be solved using a scalarization approach where the objective is a weighted sum of the objectives from individual tasks [12]. Most methods for hierarchical control consider a strict hierarchy, which is also the focus of this work. However, hybrid methods also exist that can deal with both strict and softened hierarchy [13].

Most works in the past are based on the Inverse Difference Kinematic Control (IDKC) method, where tasks are formulated as linear feedback control laws. Therefore, the corresponding Hierarchical Task IDKC (HTIDKC) problem is also linear. HTIDKC can be solved analytically by finding solutions in the null space of preceding tasks [14], which was later generalized to handle any number of tasks using a recursive algorithm [15]. Reformulated as an optimization problem, HTIDKC can also be solved with a sequence of quadratic programs (QP) in [16]. In [9], a more scalable solver was proposed, which is the state-of-the-art Hierarchical Quadratic Programming (HQP) approach.

The optimization formulation also inspires a few works on solving nonlinear hierarchical-task problems. In [5], hierarchical trajectory generation tasks are formulated as a nonlinear lexicographic optimization problem. In [17], an MPC-based hierarchical task space control method was proposed for an under-actuated and constrained robot. In [18], dynamic programming approaches were extended to address optimal control problems with hierarchical objectives. All proposed methods were either demonstrated in simulation or on a low DoF robot. Their run-time efficiency is still yet to be demonstrated as a reactive controller on high DoF robots.

In this work, we focus on solving the nonlinear HTMPC problem for sequential tasks online to enable reactive behaviours for high-dimensional robots. In [19], a real-time MPC was proposed for an industrial manipulator performing hierarchical tracking tasks. However, the core hierarchical task control problem is tackled outside the MPC controller using the HTIDKC approach. In contrast, our work aims to solve the problem directly within the MPC framework.

III. Preliminaries on Lexicographic Optimization

We give a brief introduction to lexicographic optimization. We refer readers to [20] for a complete treatment of this subject. A lexicographic optimization problem can be written as follows:

$$\operatorname{lex}\min_{\mathbf{x}\in\mathcal{X}} [f_1(\mathbf{x}), \ldots, f_L(\mathbf{x})], \tag{1}$$

where $f_1, \ldots, f_L$ is a list of tasks given in the desired order of decreasing priority. The goal is to find the optimal $\mathbf{x} \in \mathcal{X}$ that minimizes the vector objective in the sense of lexicographic order. Lexicographic order between two vectors is defined as

Definition III.1: For two vectors $\mathbf{z}^1, \mathbf{z}^2 \in \mathbb{R}^n$, $\mathbf{z}^1 <_{\text{lex}} \mathbf{z}^2$ if and only if $\exists l^* := \min\{l : z^1_l \neq z^2_l\}$, $z^1_{l^*} < z^2_{l^*}$.

Lexicographic optimality is defined as follows:

Definition III.2: A feasible solution $\hat{\mathbf{x}} \in \mathcal{X}$ is lexicographically optimal or a lexicographic solution if there is no other $\mathbf{x} \in \mathcal{X}$ such that $\mathbf{f}(\mathbf{x}) <_{\text{lex}} \mathbf{f}(\hat{\mathbf{x}})$.

Local lexicographic optimality definition can be similarly formulated by limiting the set $\{\mathbf{x} \in \mathcal{X}\}$ to a small norm ball centered at $\hat{\mathbf{x}}$, $\{\mathbf{x} \in \mathcal{X} : \|\mathbf{x} - \hat{\mathbf{x}}\| \leqslant \sigma\}$. Intuitively, the lexicographic order of two vectors depends on the first index l^* where the two vectors differ. The vector that has a smaller l^*th entry is also smaller in the sense of lexicographic order.

Lexicographic order implements a sense of hierarchy, decreasing in priority from z_0 to z_n since the order of (z^1_l, z^2_l) becomes relevant only if all the preceding entries are equal. Similarly, in (1), the order of scalar objectives in the vector also indicates their hierarchy in decreasing order from f_1 to f_L, so the optimality of f_l is considered only after the optimality of $[f_1, \ldots, f_{l-1}]$ have been established.

The interpretation of lexicographic order as a hierarchy gives rise to a common algorithm (Algorithm 1) for solving lexicographic optimization problems [20]. Scalar objective functions are optimized sequentially from f_1 to f_L as a single objective optimization problem (2). For each iteration, a new constraint is added (2b) to enforce lexicographic order by prohibiting the optimality of preceding objectives from being jeopardized while optimizing the current objective.

There are two challenges when applying Algorithm 1 to mobile manipulators: *(i)* problem (2) is nonlinear and non-convex, *(ii)* with gradient-based solvers, it is costly to determine solution uniqueness. In this work, we focus on addressing *(i)* to enable reactive behaviours for mobile manipulators executing sequential tasks while retaining local optimality. Moreover, we do not attempt to check the solution uniqueness for early termination; we will allow the algorithm to iterate through all tasks in the objective, although the solution might not be updated. To reduce redundant computation, we will manually choose the number of tasks so that their total dimension is smaller than the robot's DoF.

Algorithm 1: Lexicographic Optimization.

```
Require: Feasible set $\mathcal{X}$ and objective function $f$
1: while $l \leqslant L$ do
2:     Solve the single objective optimization problem
           $f^*_l(x) = \min_{x\in\mathcal{X}} f_l(x)$    (2a)
           s.t. $f_i(x) = f^*_i(x)$,    (2b)
           for $i = 1, \ldots, l-1$    (2c)
3:     If (2) has a unique optimal solution $\hat{x}_l$, STOP, $\hat{x}_l$ is
       the unique optimal solution
4:     If (2) is unbounded, STOP, (1) is unbounded.
5:     If $l = L$, STOP, the optimal solutions is.
           $\{x \in \mathcal{X} : f_l(x) = f^*_l(x), l = 1, \ldots, L\}$    (3)
6:     $l \leftarrow l + 1$
7: end while
9: return Set of lexicographically optimal solutions
```

IV. Methodology

Our proposed HTMPC architecture is presented in Fig. 2. Similar to typical methods, the motion planner uses separate EE and base planners and generates a sequence of task-space trajectory tracking task $[\mathcal{T}_1, \mathcal{T}_2, \mathcal{T}_3, \mathcal{T}_4, \ldots]$. At any time step, only a part of this list is given to the HTMPC module, $[\mathcal{T}_{l_o}, \ldots, \mathcal{T}_{l_o+L-1}]$, which will be updated if $\mathcal{T}_{l_o}$ is completed by incrementing l_o by 1. HTMPC controller maintains the time-ordered sequence of the given tasks and generates joint-space trajectories that coordinate the robot's all body parts toward multiple tasks. In this section, we present the HTMPC controller and show how it enables optimal and reactive behaviours for sequential mobile manipulation tasks.

A. System and Task Model

We consider a mobile manipulator with state $\mathbf{x} = [\mathbf{q}^{\mathrm{T}}\ \mathbf{v}^{\mathrm{T}}]^{\mathrm{T}}$ where $\mathbf{q} \in \mathbb{R}^n$ denotes the generalized coordinates of both the base and the arm, and $\mathbf{v} \in \mathbb{R}^m$ denotes the generalized velocity. The generalized coordinate and velocity are related by $\dot{\mathbf{q}} = \mathbf{G}(\mathbf{q})\mathbf{v}$ where $\mathbf{G}(\mathbf{q})$ is a block diagonal matrix with two components: an identity matrix for the arm and a position-dependent matrix for the mobile base [21]. We choose accelerations as the control inputs, $\mathbf{u} = \dot{\mathbf{v}}$. The feasible sets for robot state and control inputs are $\mathcal{X}$ and $\mathcal{U}$, respectively. Robot's kinematic model can be written as

$$\dot{\mathbf{x}} = \mathbf{A}(\mathbf{q})\mathbf{x} + \mathbf{B}\mathbf{u}. \tag{4}$$

where

$$\mathbf{A} = \begin{bmatrix} \mathbf{0}_{n\times n} & \mathbf{G}(\mathbf{q}) \\ \mathbf{0}_{m\times n} & \mathbf{0}_{m\times m} \end{bmatrix}, \ \mathbf{B} = \begin{bmatrix} \mathbf{0}_{n\times m} \\ \mathbf{I}_{m\times m} \end{bmatrix}. \tag{5}$$

Let point A be a point rigidly attached to the robot. Following notations in [22], its location as seen in the world frame can be

Algorithm 2: HTMPC.

```
Require: Feasible sets $(\mathcal{X},\mathcal{U})$, Initial State $\mathbf{x}_o$, , Initial
    Guess $\bar{\tilde{\mathbf{x}}}, \bar{\tilde{\mathbf{u}}}$ , and tracking cost functions $\bar{\mathcal{J}}$
1:  $\bar{\mathbf{x}}^0, \bar{\mathbf{u}}^0 \leftarrow \bar{\tilde{\mathbf{x}}}, \bar{\tilde{\mathbf{u}}}$
2:  for $l = 1, \ldots, L$ do
3:      $NLP \leftarrow$
        $constructSTMPC(\{\bar{\mathbf{x}}^{i^*}, \bar{\mathbf{u}}^{i^*}\}_{i=1}^{l-1}, \mathbf{x}_o, \mathcal{X}, \mathcal{U})$
4:      $\bar{\mathbf{x}}^{l^*}, \bar{\mathbf{u}}^{l^*} \leftarrow$
        $SQPSolve(NLP, \bar{\mathbf{x}}^{l-1}, \bar{\mathbf{u}}^{l-1}, MAX\_ITER)$
5:  end for
7:  return $\bar{\mathbf{x}}^l, \bar{\mathbf{u}}^l$
```

determined using the robot's forward kinematics, $P^A = \mathbf{f}^A(\mathbf{q})$ where $\mathbf{f}^A(\mathbf{q}) : \mathbb{R}^n \to \mathbb{R}^3$ is the forward kinematics function.

For a trajectory tracking task $\mathcal{T}$, the robot needs to follow a desired reference signal $\mathbf{r}(t) : [0, T] \to \mathbb{R}^s$ with a point on its body where t is the control time. We assume reference trajectories are specified in vector space, and the distance between $\mathbf{P}$ and $\mathbf{r}$ is

$$\text{dist}\,(\mathbf{P}, \mathbf{r}) = \|\mathbf{P} - \mathbf{r}\| = \|\mathbf{f}(\mathbf{q}) - \mathbf{r}\| := \|\mathbf{e}\|\,. \tag{6}$$

B. Hierarchical-Task MPC

We now present our HTMPC controller for the sequential trajectory tracking problem. At each control time t, the controller solves a lexicographic optimization problem for a pair of optimal state and control sequence, $\bar{\mathbf{x}}^*$ and $\bar{\mathbf{u}}^*$, over a prediction horizon N steps into the future at a discretization time step Δt. The lexicographic optimization problem is

$$\operatorname{lex}\min_{\bar{\mathbf{x}},\bar{\mathbf{u}}}\ [\mathcal{J}_1, \mathcal{J}_2 \cdot \cdots, \mathcal{J}_L] \tag{7a}$$

$$\text{s.t. } \mathbf{x}_{k+1} = \mathbf{A}(\mathbf{q}_k)\mathbf{x}_k + \mathbf{B}\mathbf{u}_k \tag{7b}$$

$$\mathbf{sd}(\mathbf{x}_k) \geqslant \boldsymbol{\delta}_{safe} \tag{7c}$$

$$\mathbf{x}_k \in \mathcal{X}, \mathbf{u}_k \in \mathcal{U},\ \forall k = 0, 1, \ldots, N-1 \tag{7d}$$

$$\mathbf{x}_0 = \mathbf{x}_o. \tag{7e}$$

where $\mathbf{x}_o = \mathbf{x}(t)$ is the current state of the robot. State and control input at prediction step k are $\mathbf{x}_k = \mathbf{x}(t + k\Delta t)$, $\mathbf{u}_k = \mathbf{u}(t + k\Delta t)$, respectively.

Similar to a typical MPC controller, HTMPC has constraints for motion model (7b), collision avoidance (7c), state (7d) and control input as well as state initialization (7e). As in [23], we cover the robot with spheres and require the signed distance between theses spheres $\mathbf{sd}(\mathbf{x}_k)$ to be greater than a safe distance $\boldsymbol{\delta}_{safe}$. Instead of a scalar cost function, HTMPC has a vector of scalar cost functions (7a). Each scalar cost function represents the accumulated tracking error for a task in the sequence and is defined as

$$\mathcal{J} = \frac{1}{2}\left\{\sum_{k=0}^{N-1} \|\mathbf{e}_k\|_{\mathbf{Q}_k}{}^2 + \|\mathbf{e}_N\|_{\mathbf{P}}{}^2\right\}, \tag{8}$$

where matrices $\mathbf{Q}_k$ and $\mathbf{P}$ are diagonal positive semi-definite (PSD) weight matrices for stage and terminal cost with non-negative diagonal terms. Task index l and dependency on $\mathbf{x}$ are dropped in (8) for simplicity. HTMPC enforces the time-ordered sequence in the tasks by establishing a hierarchy in their cost functions. In particular, the cost functions are arranged into a vector in the same order as in the task sequence; the cost function $\mathcal{J}_l$ corresponds to the l^{th} task in the given list, $\mathcal{T}_{l_o-1+l}$, for which we will use $\mathcal{T}_l$ going forward. Lexicographic optimality requires that a task $\mathcal{T}_l$ is tackled only when optimal tracking has been established for preceding tasks; hence, the time-ordered sequence is maintained.

C. Single-Task MPC: A Building Block for Solving HTMPC

We follow Algorithm 1 to solve the HTMPC problem (7). In iteration l, the following scalar optimization problem is solved to optimize $\mathcal{T}_l$:

$$\min_{\bar{\mathbf{x}},\bar{\mathbf{u}}}\ \mathcal{J}_l + \mathcal{J}_{\mathit{eff}} \tag{9a}$$

$$\text{s.t. (7b), (7c), (7d), (7e)} \tag{9b}$$

$$h_i(\bar{\mathbf{x}}, \bar{\mathbf{u}}; \bar{\mathbf{x}}^{i^*}, \bar{\mathbf{u}}^{i^*}) \leqslant 0, \forall i = 1, \ldots, l-1\,. \tag{9c}$$

We denote the optimal state and control sequences to (9) as $\bar{\mathbf{x}}^{l^*}, \bar{\mathbf{u}}^{l^*}$. The parameters in (9c) $\bar{\mathbf{x}}^{i^*}, \bar{\mathbf{u}}^{i^*}$ are the optimal solution to $\mathcal{T}_i$ obtained in iteration $l = i$.

Compared to (2), we made two changes to the formulation. First, we added a control effort cost function to the objective:

$$\mathcal{J}_{\mathit{eff}} = \frac{1}{2}\left\{\|\bar{\mathbf{x}}\|_{\bar{\mathbf{S}}}^2 + \|\bar{\mathbf{u}}\|_{\bar{\mathbf{R}}}^2 + \|\Delta\bar{\mathbf{u}}\|_{\bar{\mathbf{W}}}^2\right\}, \tag{10}$$

where $\bar{\mathbf{S}}$, $\bar{\mathbf{R}}$ and $\bar{\mathbf{W}}$ are diagonal weight matrices for robot state $\bar{\mathbf{x}}$, control inputs $\bar{\mathbf{u}}$, and time difference of control inputs $\Delta\bar{\mathbf{u}}$ with non-negative terms of appropriate dimensions. Similar to other MPC methods, $\mathcal{J}_{\mathit{eff}}$ encourages smooth motion and improves numerical stability. Similar to the HTIDKC method adding regularization terms, this could also improve numerical stability in case of singularity [9].

Second, we reformulated the lexicographic optimality constraint as (9c), for which we consider two formulations. The first formulation closely resembles (2b):

$$\mathcal{J}_i(\bar{\mathbf{x}}, \bar{\mathbf{u}}) \leqslant \mathcal{J}_i(\bar{\mathbf{x}}^{i^*}, \bar{\mathbf{u}}^{i^*}). \tag{11}$$

However, the equality has been replaced with an inequality to reflect the fact that $\bar{\mathbf{x}}^{i^*}, \bar{\mathbf{u}}^{i^*}$ no longer minimizes the accumulated tracking error but the sum of both tracking error and control effort. Similar to [18], we also proposed to approximate (11) with decoupled constraints

$$|e_{kp}| \leqslant |e_{kp}^{i^*}|, \text{ for } k = 0, \ldots, N,\ p = 1, \ldots, s, \tag{12}$$

directly limiting the tracking error $\mathbf{e}$ at prediction time step k along task space dimension p with its optimal value $\mathbf{e}_k^{i^*} = \mathbf{f}(\mathbf{q}_k^{i^*}) - \mathbf{r}_k$. Note that the subscript i for $\mathcal{T}_i$ is omitted in (12) for simplicity. The decoupled constraints (12) is an inner approximation of (11), which, we found, offers a better convergence rate when solving (9).

D. Solving HTMPC

We present an algorithm for solving HTMPC in Algorithm 2. Overall, we follow the sequential approach outlined in Algorithm 1 but without the uniqueness test. In each iteration, an Single-Task MPC (STMPC) problem (9) is constructed using the initial state, feasible sets for robot constraints, and solutions from previous iterations for lexicographic constraints. We choose to solve STMPC, a nonlinear program (NLP), using the Sequential Quadratic Programming (SQP) approach. SQP solves NLPs with a sequence of QPs via incremental quadratization with a maximum iteration MAX_ITER. Each QP gives a local update direction along which the best step size is determined based on convergence and feasibility conditions. The initial guess for the SQP, $\bar{\mathbf{x}}^{l-1}$ and $\bar{\mathbf{u}}^{l-1}$, is either provided externally for the first iteration or solutions from the previous iteration.

Optimality of Algorithm 2 depends on the STMPC solutions. SQP approach finds a locally optimal solution to STMPC if converged [24]. Therefore, the lexicographic optimality constraints (9c) only enforce lexicographic order in a local region. As a result, solutions by the proposed Algorithm 2 are locally optimal. If STMPC reaches MAX_ITER before converges, the incremental update is still an admissible step towards a local minima of (7) [5].

We share some notes on implementing HTMPC on a real system. The inequality constraints in STMPC are formulated as hard constraints. However, when running on a real system, disturbances, sensor noises, and state estimation errors might render the MPC problem infeasible, leading to a complete failure if no remedies are taken. In recent years, it has become more common to soften these constraints in implementation. One method is to incorporate them into cost functions via barrier functions. It is also possible to relax the inequality constraints with slack variables whose norm is minimized in the objective. The trade-off between minimizing the original cost function and constraints violation is controlled indirectly by tuning weight parameters. We used the relaxed log barrier function method for state and control constraints (7d) as well as collision constraints (7c).

For the lexicographic optimality constraints (9c), we took an alternative approach to have quantitative control over their relaxation. More specifically, in the line search step of SQP, instead of finding a step size that strictly satisfies the lexicographic constraints, we allow violations within a predefined range. That is, a step size is feasible if it satisfies

$$h_i \leqslant \delta_i, \ \forall i = 1, \dots, l-1, \tag{13}$$

where $\delta_i \in \mathbb{R} > 0$ is the tolerance for $\mathcal{T}_i$. (13) allows us to quantify the maximal tracking error that preceding tasks can compromise to improve subsequent tasks at each control time step. This becomes useful in practice where different tasks and applications require different levels of control accuracy. In the next section, we will show that this can be leveraged to improve hierarchical task performance.

V. Experiments

Our experimental platform is a 9 DoF mobile manipulator platform consisting of a holonomic Ridgeback mobile base and a 6 DoF UR10 arm (Fig. 1). The pose of the mobile base is measured by an external motion tracking system, whereas the EE pose is computed from the arm's joint positions measured by its internal joint encoders. The proposed HTMPC is implemented using the *CasAdi* framework in Python [25]. A standard SQP procedure in [24] with the proposed back-tracking line search was implemented for STMPC with QP solver from *Gurobi* [26]. For all results in this section, we used $\Delta t = 0.1$ s, $N = 10$, MAX_ITER $= 1$. The average compute time of HTMPC is 63 ms, so its control frequency was set at 10 Hz. All results are experimental data except for those in Section V-B. Videos of all experimental results can be found at http://tiny.cc/htmpc.

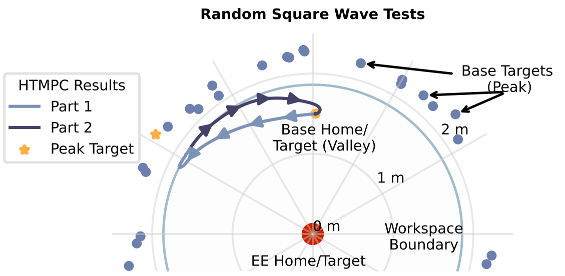

Fig. 3. Illustration of 25 random square wave tests plotted in end effector frame using polar coordinates. The hierarchical tasks are, $\mathcal{T}_0$ minimizing constraints violation, $\mathcal{T}_1$ holding its end effector in home position, and $\mathcal{T}_2$ base tracking a square wave trajectory. The square wave trajectories have a duration of 16 s that peaks during $t = 0 \sim 8$ s (Part 1) and bottoms during $t = 8 \sim 16$ s (Part 2). Valley target is the same for all cases which is the robot's home position. Peak targets are randomly selected outside the robot's workspace. An example base trajectory executed by HTMPC is provided with arrows indicating the direction of motion. Its corresponding peak target is marked as a yellow star. Note that Fig. 7 is plotted in the same coordinate to show the correlation between the base tracking performance and the base targets' position.

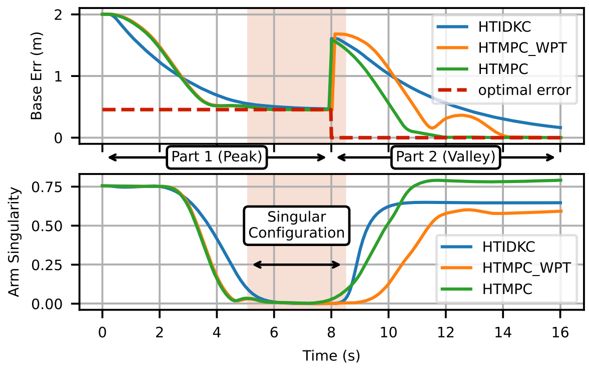

Fig. 4. Experimental results of the three competing methods for the example shown in Fig. 3. Since the peak target is not reachable, all three methods converge to a non-zero base error in Part 1 when the robot arm becomes fully extended and singular. For Part 2, only the proposed methods HTMPC and HTMPC_WPT are converged to its valley target. HTIDKC did not converge because the parameters were optimized for Part 1.

A. Hierarchical Random Square Wave Tests

To see how well HTMPC enforces the time-ordered sequence with hierarchy, we evaluated it with 25 randomly generated tests with three-level hierarchical tasks illustrated in Fig. 3. The two tracking tasks represent the sub-sequence given to the controller as part of the process that fulfills the high-level user request. The test is designed to see how controllers can handle task changes, singular configurations, and variations in reference trajectories. We chose square wave trajectories for the base to test the controller's capability in reacting to changes in the subsequent task initiated by the upper-level planning module.

An example base trajectory executed by HTMPC is shown in Fig. 3 as two parts. The base moves from its home position to reach the peak target (yellow star) (Part 1) before moving back to its home position (Part 2). The base only moves as far as its workspace boundary to avoid compromising the EE task. The corresponding tracking error is shown in Fig. 4. In Part 1, the base tracking error converges to its optimal steady-state error as it comes to its workspace boundary with the arm fully extended. In Part 2, the base tracking error should converge to zero since the valley target is reachable.

TABLE I
HTMPC EE TRACKING ERROR WITH DIFFERENT LEXICOGRAPHIC OPTIMALITY CONSTRAINTS (MM)

Eq	Low δ		High δ	
	avg	std	avg	std
Baseline (11)	3.24	0.56	6.25	2.73
Proposed (12)	**3.07**	**0.73**	**3.42**	**1.55**

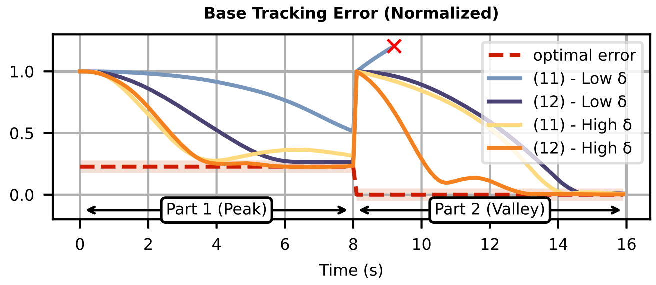

Fig. 5. HTMPC normalized base tracking error with different lexicographic constraints. The red cross indicates that the base tracking has diverged and is removed from the plot. A desired steady state error bound of 5% is shaded in red. Formulation (12) outperforms (11) with shorter convergence time for both tolerance levels. Increasing the constraint tolerance improves base tracking performance at the cost of the EE task (Table I).

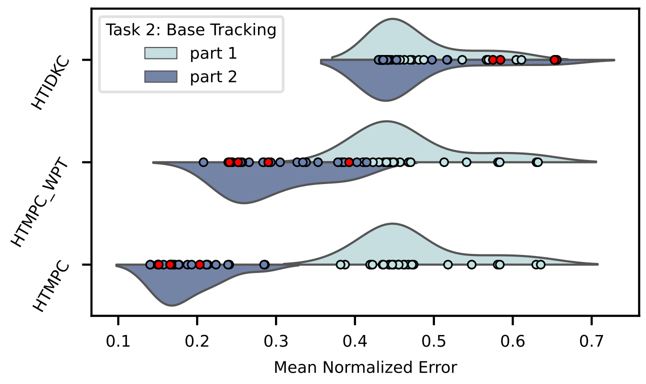

Fig. 6. Violin plot showing the hierarchical task results of the three competing methods for the 20 random square wave tests. Both distributions (shaded area) and individual data points (dots) are shown. Results are presented separately for Part 1 (light blue) and Part 2 (dark blue). Part 2 results for the singularity impacted cases (Fig. 7) are highlighted (red). Parameters are designed so that all methods have similar performance for $\mathcal{T}_0$, $\mathcal{T}_1$, and Part 1 of $\mathcal{T}_2$. HTMPC achieves the best control performance in Part 2 of $\mathcal{T}_2$ with a 42% improvement over the HTIDKC approach. Therefore, HTMPC achieves a lower cost of the hierarchical tracking problem in the sense of lexicographic order as per Definition 3.2.

B. Lexicographic Optimality Constraints

We tested the two lexicographic constraint formulations on the example square wave test shown in Fig. 3 and presented the results in Table I and Fig. 5. The proposed relaxation method (13) was also implemented, and two levels of tolerance were tested. The tolerance value δ was set at a comparable level for both formulations (1 cm EE tracking error for High δ, and 1 mm for Low δ) [1]. Our proposed base tracking error is reported separately for Part 1 and Part 2 in Fig. 5. Both are normalized by

[1]Note that both formulations with low tolerance failed to maintain e_{EE} within the desired error 1 mm error bound. The main reason is that lexicographic constraints are imposed on the open-loop predicted trajectory. Therefore, closed-loop tracking error is not guaranteed.

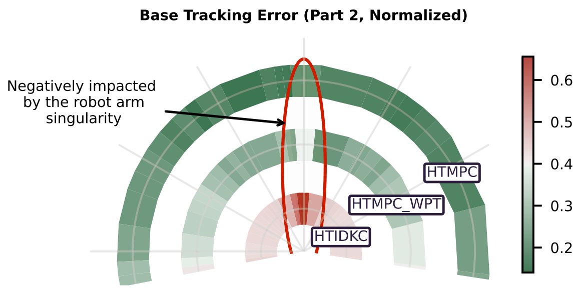

Fig. 7. Radial heat map showing the spatial correlation between the normalized Part 2 base tracking error and peak targets. As in Fig. 3, this plot is in polar coordinates where the angular component is the relative direction of base peak target to the end effector home position. Circled in red is the region where singularity plays a major role in tracking performance. HTMPC outperforms the other two methods in dealing with singularity.

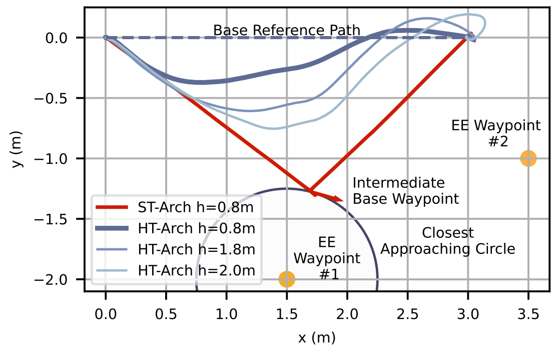

Fig. 8. Mobile manipulator base path comparison of a typical single-task (ST-Arch) and our proposed hierarchical-task (HT-Arch) architectures for a sequence of two EE waypoint tasks. The EE waypoint #1 were set at different heights. The intermediate base waypoint for the ST-Arch is determined following [10]. Compared to ST-Arch, the proposed HT-Arch approach (HT-Arch h = 0.8 m) optimally positions the robot base where the EE waypoint #1 just becomes reachable.

the initial error of each part. Our proposed formulation (12) leads to improved tracking performance for both tasks mainly because it better approximates the nonlinear feasible set after linearization, permitting larger step sizes during line search. For both formulations, increasing tolerance can improve the secondary task but at the cost of the primary task. However, compromise is transient, so we only see a non-negligible increase to e_{EE} standard deviation.

C. HTMPC Versus HTIDKC

We compared the tracking performance of HTMPC with HTIDKC using the 25 randomly sampled tests. HTIDKC is implemented following formulations in [9]. The three-level tasks were precisely the same for both approaches with the only task difference being that an additional jerk bound was implemented for HTMPC to encourage smooth motions when running on a real robot, which was not necessary for HTIDKC. Regularization was also implemented in cost function for HTIDKC to deal with singularity. We used the cascaded QP approach with regularization terms included for singularity [9] to solve HTIDKC and implemented it in *CasAdi* with *Gurobi* QP solver. The compute time for HTIDKC is 17 ms on average, so its control frequency was set at 50 Hz. To simplify lexicographic comparisons, the two controllers were tuned to have similar performance for the top two tasks and Part 1 for the base tracking

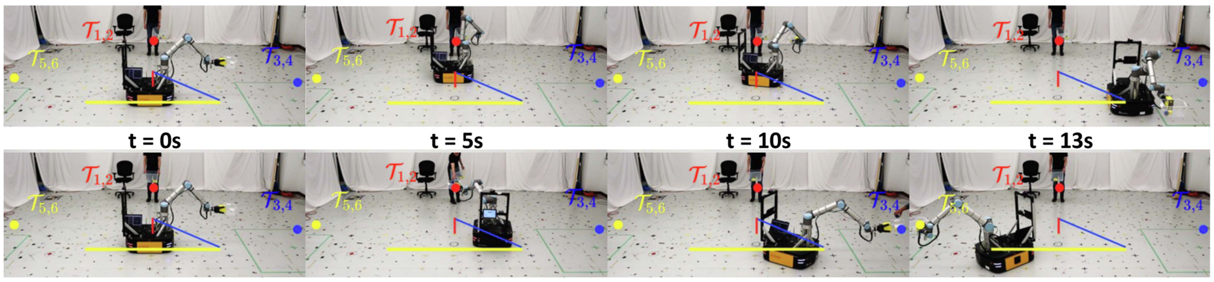

Fig. 9. Robot performing a sequence of delivery tasks to three people. *Top figure* shows our robot running an arm-base decoupled single-task control architecture. *Bottom figure* shows robot running a our proposed HTMPC control architecture. Our proposed architecture is more efficient in executing sequential mobile manipulation tasks by allowing the base to move its next delivery target while holding EE in place.

task; therefore, we only need to compare Part 2 to determine their lexicographic optimality.

In addition to HTIDKC, we also compared with a second baseline, HTMPC_WPT. This method is HTMPC receiving the square wave trajectory as two separate waypoints instead of one trajectory. As a result, similar to HTIDKC, HTMPC_WPT does not foresee the task change at t = 8 s.

Tracking error is plotted over control time in Fig. 4 for the example in Fig. 3. A violin plot showing a fitted distribution for the normalized base error is shown in Fig. 6. Part 1 is designed to have a similar performance. For Part 2, HTIDKC has the worst tracking performance of all three methods—it does not have any improvements over Part 1 despite the fact that the valley target is within reach and a lower mean error can be achieved. In contrast, both HTMPC methods have a lower tracking error for Part 2 mainly because the controller parameters were optimized for Part 1 and HTIDKC does not generalize to Part 2 as well as the optimization-based HTMPC controller. Parameter scheduling is required for HTIDKC to have comparable results to HTMPC. HTMPC also slightly outperforms HTMPC_WPT, which can be attributed to the predictive nature of MPC. When receiving one complete trajectory instead of two separate waypoints, HTMPC foresees the coming step in the trajectory and reacts earlier to have an overall lower tracking error. As seen in Fig. 4, HTMPC converges faster than HTMPC_WPT in Part 2 while incurring negligible error in Part 1.

Normalized base tracking error (Part 2) is shown as a radial heat map in Fig. 7 to illustrate its spatial correlation with peak targets. All three methods deteriorate as the base peak target, valley target and EE target become colinear in x-y plane (highlighted in red in both Figs. 6 and 7). In these cases, the optimal base motion demands aggressive robot arm maneuvers with high joint velocities to prioritize the EE task when the robot arm is near its singular configuration. Of all three methods, HTMPC has the least decline in performance due to better regularization of both the joint velocity and acceleration and better constraint handling via prediction and line search.

D. *Hierarchical-Task Versus Baseline Control Architectures*

In this section, we compare the optimality of our proposed HTMPC-based hierarchical-task control architecture (HT-Arch) with a typical single-task architecture (ST-Arch) as a baseline, the arm-base decoupled architecture similar to [2]. The task sequence were given to the controller as described in Section IV, and we chose $L = 1$ for ST-Arch and $L = 2$ for HT-Arch. Although we only implemented the single-task method, the multi-task architecture [10] would have given similar results for the first task.

The first task is illustrated in Fig. 8, where the robot needs to visit two EE waypoints in a sequence. Although EE waypoints are the same for both approaches, base reference paths were optimized differently for the underlying controller. With a decoupled arm and base controller, base reference path for ST-Arch needs to go through an intermediate waypoint at a predefined distance from the first EE target before reaching the second one. In contrast, since HTMPC can perform redundancy resolution using robot's kinematics, it is sufficient to have a path going straight towards the second EE target for HT-Arch. Actual base paths executed by the controllers are also different. ST-Arch closely follows the given reference to reach the EE target whereas HT-Arch deviates to optimally position its base while holding its EE on target. Additionally, HTMPC in HT-Arch can automatically adapt its base path to EE targets at different heights, whereas ST-Arch relies on the motion planner to adjust the intermediate base waypoint for its controller based on heuristics [11].

We also compared the two architectures on time efficiency with a long horizon task where the robot needs to perform a sequence of delivery tasks to three people for four rounds. In this test, both the decoupled arm and base controller and HTMPC were given the same base and EE plans. Leveraging HTMPC's multi-tasking nature for sequential tasks, our proposed HT-Arch approach executes the tasks 2.3 times faster than the ST-Arch approach (Fig. 9).

We also introduced changes to the robot delivery task to demonstrate the task-space reactivity of our proposed architecture. In the first scenario, the robot is forced to hold its EE in place until the person picked out the ball whose timing is unknown to the controller. HTMPC allowed the base to continue moving toward its next target as usual but stopped the base immediately before compromising the EE task. Moreover, HTMPC automatically updated the base position as the next delivery target changes location. We also showed that our proposed architecture can quickly react to an unexpected shift in the current task via fast replanning and efficient hierarchical control. Both tests can be found in the accompanied video and also at http://tiny.cc/htmpc.

VI. Conclusion

We presented the HTMPC framework for solving sequential mobile manipulation tasks. The framework centers around the HTMPC controller that enforces the time-ordered sequence of tasks with a hierarchy in their trajectory tracking cost functions. We demonstrated that HTMPC outperforms the state-of-the-art HTIDKC approach in hierarchical trajectory tracking tasks. We also presented a hierarchial-task control architecture optimized for HTMPC to leverage the robot's redundancy for sequential tasks. We demonstrated that our proposed architecture has improved efficiency and reactivity in executing sequential tasks compared to the existing architectures. In future works, we wish to investigate alternative regularization and lexicographic constraint formulations to improve our proposed HTMPC algorithm.

REFERENCES

[1] M. Ahn and A. Zeng, "Do as I can, not as I say: Grounding language in robotic affordances," Aug. 2022, *arXiv: 2204.01691*.

[2] J. Carius, M. Wermelinger, B. Rajasekaran, K. Holtmann, and M. Hutter, "Deployment of an autonomous mobile manipulator at MBZIRC," *J. Field Robot.*, vol. 35, no. 8, pp. 1342–1357, 2018.

[3] J. Haviland, N. Sünderhauf, and P. Corke, "A holistic approach to reactive mobile manipulation," *IEEE Robot. Automat. Lett.*, vol. 7, no. 2, pp. 3122–3129, Apr. 2022.

[4] J. Pankert and M. Hutter, "Perceptive model predictive control for continuous mobile manipulation," *IEEE Robot. Automat. Lett.*, vol. 5, no. 4, pp. 6177–6184, Oct. 2020.

[5] Y. Tazaki and T. Suzuki, "Constraint-based prioritized trajectory planning for multibody systems," *IEEE Trans. Robot.*, vol. 30, no. 5, pp. 1227–1234, Oct. 2014.

[6] S. Zimmermann, R. Poranne, and S. Coros, "Go fetch! - dynamic grasps using Boston dynamics spot with external robotic arm," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2021, pp. 4488–4494.

[7] S. Thakar, P. Rajendran, V. Annem, A. Kabir, and S. Gupta, "Accounting for part pose estimation uncertainties during trajectory generation for part pick-up using mobile manipulators," in *Proc. Int. Conf. Robot. Automat.*, 2019, pp. 1329–1336.

[8] S. Thakar, P. Rajendran, A. M. Kabir, and S. K. Gupta, "Manipulator motion planning for part pickup and transport operations from a moving base," *IEEE Trans. Automat. Sci. Eng.*, vol. 19, no. 1, pp. 191–206, Jan. 2022.

[9] A. Escande, N. Mansard, and P.-B. Wieber, "Hierarchical quadratic programming: Fast online humanoid-robot motion generation," *Int. J. Robot. Res.*, vol. 33, no. 7, pp. 1006–1028, Jun. 2014.

[10] B. Burgess-Limerick, C. Lehnert, J. Leitner, and P. Corke, "An architecture for reactive mobile manipulation on-the-move," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2023, pp. 1623–1629.

[11] B. Burgess-Limerick, C. L. J. Leitner, and P. Corke, "Enabling failure recovery for on-the-move mobile manipulation," in *Proc. Robot Execution Failures Failure Manage. Strategies Wrokshop Int. Conf. Robot. Automat.*, 2023.

[12] A. S. Sathya, G. Pipeleers, W. Decré, and J. Swevers, "A weighted method for fast resolution of strictly hierarchical robot task specifications using exact penalty functions," *IEEE Robot. Automat. Lett.*, vol. 6, no. 2, pp. 3057–3064, Apr. 2021.

[13] N. Dehio and J. J. Steil, "Dynamically-consistent generalized hierarchical control," in *Proc. Int. Conf. Robot. Automat.*, 2019, pp. 1141–1147.

[14] H. Hanafusa, T. Yoshikawa, and Y. Nakamura, "Analysis and control of articulated robot arms with redundancy," *IFAC Proc. Vol.*, vol. 14, no. 2, pp. 1927–1932, Aug. 1981.

[15] B. Siciliano and J.-J. Slotine, "A general framework for managing multiple tasks in highly redundant robotic systems," in *Proc. 5th Int. Conf. Adv. Robot. 'Robots Unstructured Environments*, 1991, pp. 1211–1216.

[16] O. Kanoun, F. Lamiraux, and P.-B. Wieber, "Kinematic control of redundant manipulators: Generalizing the task-priority framework to inequality task," *IEEE Trans. Robot.*, vol. 27, no. 4, pp. 785–792, Aug. 2011.

[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 *Proc. IEEE 59th Conf. Decis. Control*, 2020, pp. 5942–5949.

[18] F. Romano, A. D. Prete, N. Mansard, and F. Nori, "Prioritized optimal control: A hierarchical differential dynamic programming approach," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2015, pp. 3590–3595.

[19] J. Lee, M. Seo, A. Bylard, R. Sun, and L. Sentis, "Real-time model predictive control for industrial manipulators with singularity-tolerant hierarchical task control," in *Proc. Int. Conf. Robot. Automat.*, 2023, pp. 12282–12288, doi: 10.1109/ICRA48891.2023.10161138.

[20] M. Ehrgott, *Multicriteria Optimization*. Berlin, Germany: Springer-Verlag, 2005.

[21] B. Siciliano, L. Sciavicco, L. Villani, and G. Oriolo, *Robotics: Modelling, Planning and Control*, 1st ed. Berlin, Germany: Springer, 2008.

[22] R. Tedrake, "Robotic manipulation," 2022. [Online]. Available: http://manipulation.mit.edu

[23] M. Gaertner, M. Bjelonic, F. Farshidian, and M. Hutter, "Collision-free MPC for legged robots in static and dynamic scenes," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2021, pp. 8266–8272.

[24] S. P. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge, U.K.: Cambridge Univ. Press, 2004.

[25] J. A. E. Andersson, J. Gillis, G. Horn, J. B. Rawlings, and M. Diehl, "CasADi: A software framework for nonlinear optimization and optimal control," in *Mathematical Programming Computation*, vol. 11. Berlin, Germany: Springer, 2019, pp. 1–36.

[26] G. Optimization LLC, "Gurobi optimizer reference manual," 2023. [Online]. Available: https://www.gurobi.com