2022 30th Mediterranean Conference on Control and Automation (MED)
June 28 - July 1, 2022. Athens, Greece

2022 30th Mediterranean Conference on Control and Automation (MED) | 978-1-6654-0673-4/22/$31.00 ©2022 IEEE | DOI: 10.1109/MED54222.2022.9837159

Real-Time Motion Generation for Mobile Manipulators via NMPC with Balance Constraints

Spyridon G. Tarantos Giuseppe Oriolo

Abstract— **We present a novel real-time motion generation approach for mobile manipulators which maintains balance even when the robot is called to execute aggressive motions. The proposed approach is based on Nonlinear Model Predictive Control (NMPC) and uses the robot full dynamics as prediction model. Robot balance is maintained by enforcing a constraint that restricts the feasible set of robot motions to those generating non-negative moments around the edges of the support polygon. This balance constraint, inherently nonlinear, is linearized using the NMPC solution of the previous iteration. In this way we facilitate the solution of the NMPC and we achieve real-time performance without compromising robot safety. We validate our approach in scenarios of increasing difficulty and compare its performance with two other methods from the literature. The simulation results show that our method can generate motions that maintain balance in challenging situations where the other techniques fail.**

I. Introduction

Mobile manipulators (MMs) are increasingly used in applications due to their inherent versatility, as they combine the mobile base mobility and the manipulator dexterity. Depending on the application, their structure and size varies. MMs with a small base are suitable for narrow and cluttered spaces, like industrial or even domestic environments. However, as the size of the base decreases, one has to focus on the robot balance which becomes more fragile, especially under the execution of aggressive motions.

In the literature there are many attempts to evaluate and prevent the MM loss of balance. The motion planning approach in [1] maintains the MM balance when the base is stationary. The *Force-Angle* measure [2], [3] predicts and prevents tip-over instabilities when the whole robot is in motion. The extensively used in humanoids Zero Moment Point (ZMP) [4] is applied to MMs in [5]–[7] neglecting though the inertia effect of the robot bodies. In order to reduce the computational effort [8] uses approximations of the ZMP position gradient and Hessian matrix, while in [9] such approximations are avoided using appropriate recursive algorithms. In [10], [11] the Stability Twist Constraint (STC) is used in an inverse kinematics solver implemented as a Quadratic Program (QP) constraining the support wrench on the robot base to produce non-positive power with an imaginary twist along each axis of the robot support polygon. Similarly, [12] introduces an algorithm that adjusts the robot motion in order to recover from a loss of balance considering the moment around the support polygon edges. Note that in [8]–[12] a high-level planner is required.

The authors are with the Dipartimento di Ingegneria Informatica, Automatica e Gestionale, Sapienza Università di Roma, via Ariosto 25, 00185 Roma, Italy. E-mail:{tarantos,oriolo}@diag.uniroma1.it

An increasingly used tool for motion generation is the NMPC. The NMPC solves an Optimal Control Problem (OCP) at each control cycle and obtains the control inputs that will be applied to the dynamic system. An application of linear MPC to an omnidirectional MM is presented in [13]. The kinematic model of an MM moving under nonholonomic constraints is considered in the NMPC of [14] where a Sequential Linear Quadratic (SLQ) algorithm [15] is used for the solution of the OCP. However, no inequality constraints like control input limits and collision avoidance constraints are considered. The SLQ algorithm is used also in [16] including inequality constraints through relaxed barrier functions and, unlike [13] and [14] where the robot balance is not considered, here the authors introduce a balance constraint using an approximation of the ZMP position that ignores the effect of the dynamics under the assumption of slow motions. Nevertheless, an approximation like this can be dangerous when the robot performs aggressive motions, since in this case the effect of the dynamics is significant.

In this work we present an NMPC suitable for MMs called to execute tasks that require aggressive motions (i.e., high accelerations). Unlike the aforementioned works ([13], [14], [16]) we enforce kinodynamic feasibility through the use of the robot dynamics as a prediction model and via input and state constraints. We also include state constraints to guarantee collision free motions. The relatively small base of the considered robot and the required fast motions dictate the use of a balance constraint that fully considers the effect of the dynamics. We adopt the STC constraint for balance, which requires the moments exerted by the robot around the support polygon edges to remain non-negative. For the solution of the OCP we use a Sequential Quadratic Programming (SQP)-based Real-Time Iteration (RTI) approach [17] implemented within ACADO [18]. To avoid the memory overhead coming with software like ACADO, that are based on the symbolic representation of the OCP, we simplify the inherently nonlinear balance constraint. We do so by linearizing the constraint using the solution from the previous iteration of the NMPC, considering it as a satisfying approximation of the current solution. The proposed NMPC is compared with the approaches presented in [10] and [16]. The results show that the proposed method outperforms the compared ones by effectively generating feasible motions in real-time in a series of challenging scenarios that require the execution of aggressive motions.

The paper is organised as follows. The considered problem is formulated in Section II. In Section III we define the robot equations of motion and we present the formulas for the

computation of the contact forces. In Section IV we outline the proposed NMPC approach and in Section V we introduce the considered balance constraint. Simulation results for the considered MM are offered in Section VI. Finally, some concluding remarks are offered in Section VII.

II. Problem Formulation

Consider a mobile manipulator whose configuration $\boldsymbol{q}$ takes values in an n-dimensional space. The robot operates in a 3-dimensional workspace $\mathcal{W}$, populated by static and/or dynamic obstacles. The volume occupied by the robot is denoted by $\mathcal{R}(\boldsymbol{q}) \subset \mathcal{W}$. By $\mathcal{O}_j(t) \subset \mathcal{W}$ we denote the volume occupied by the j-th obstacle at time t $(j = 1, 2, ...)$. The robot is assigned a task in terms of a vector $\boldsymbol{y} \in \mathcal{Y}$, which describes the pose (position and orientation) of the end-effector and is related to the robot configuration via a forward kinematics map $\boldsymbol{y} = \boldsymbol{\sigma}(\boldsymbol{q})$. The task is assigned as a desired end-effector trajectory $\boldsymbol{y}_d(t)$ with $t \in [0, t_f]$ where t_f the task duration.

We wish to generate in real-time a motion that:

1) starts from the robot initial configuration $\boldsymbol{q}(0) = \boldsymbol{q}_s$ and tracks the assigned end-effector task as accurately as possible;
2) is kinodynamically feasible, in the sense that it is consistent with the robot dynamic model and respects existing constraints on joint and velocity limits and control input limits (e.g., torque bounds);
3) maintains the robot balance, in the sense that all wheels maintain contact with the ground;
4) avoids collisions between the robot and the obstacles as well as self-collisions.

Clearly, *exact* tracking of the end-effector task is not required, because the safety requirements 2-4 take priority.

III. Modeling

We consider an MM consisting of a general wheeled mobile base, carrying a manipulator with n_m joints (see Fig. 1). We assume that the robot moves on an horizontal ground, which is generally the case for wheeled MMs especially for indoor applications, and that its n_w wheels are in point contact with the ground. For the actuated wheels we assume that the ground-wheel friction is sufficient to prevent slippage while the rest of the wheels do not affect significantly the robot motion and they only contribute to the robot balance.

In order to represent the pose of the mobile base we consider 6 fictitious joints that connect the world frame $\mathcal{F}_w$ with a frame $\mathcal{F}_b$ that is attached to the mobile base. The joints are arranged as follows: 3 prismatic joints along the x_w, y_w and z_w axes followed by 3 revolute joints with axes parallel to the *yaw*, *pitch* and *roll* axes of the mobile base (see Fig. 1). Note that this arrangement is not unique. We denote the base configuration by $\boldsymbol{q}_b = \left(x_b, y_b, z_b, \theta_z, \theta_y, \theta_x\right)$ and the manipulator configuration by $\boldsymbol{q}_m = \left(\theta_1, ..., \theta_{n_m}\right)$. So the robot configuration is $\boldsymbol{q} = \left(\boldsymbol{q}_b, \boldsymbol{q}_m\right)$ and $n = 6 + n_m$.

The constraints enforced by the contact with the ground (including the pure rolling constraint) can be expressed in Pfaffian form [19] as $\boldsymbol{A}^T(\boldsymbol{q})\dot{\boldsymbol{q}} = \boldsymbol{0}$.

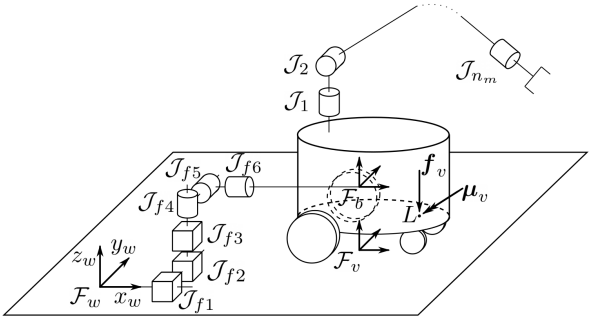

Fig. 1. A mobile manipulator consisting of a wheeled mobile base carrying a n_m joint manipulator.

The robot dynamics in the Lagrange formulation are:

$$\boldsymbol{B}(\boldsymbol{q})\ddot{\boldsymbol{q}} + \boldsymbol{n}(\boldsymbol{q}, \dot{\boldsymbol{q}}) = \boldsymbol{S}(\boldsymbol{q})\boldsymbol{u} + \boldsymbol{A}(\boldsymbol{q})\boldsymbol{\lambda}, \tag{1}$$

being $\boldsymbol{B}(\boldsymbol{q}) \in \mathbb{R}^{n \times n}$ the inertia matrix, $\boldsymbol{n}(\boldsymbol{q}, \dot{\boldsymbol{q}}) \in \mathbb{R}^n$ the vector of velocity and gravitational terms, $\boldsymbol{u} \in \mathbb{R}^{n_u}$ the vector of generalized forces applied by the n_u robot actuators, $\boldsymbol{S}(\boldsymbol{q}) \in \mathbb{R}^{n \times n_u}$ the matrix that maps the actuator forces to forces performing work on the generalized coordinates, and $\boldsymbol{A}(\boldsymbol{q})\boldsymbol{\lambda}$ the vector of the forces exerted to the robot by its contact with the ground expressed at the generalized coordinates level, with $\boldsymbol{\lambda}$ being the vector of contact forces.

Due to the robot motion on the horizontal ground, the coordinates z_b, θ_y and θ_x, which correspond to the joints $\mathcal{J}_{f3}$, $\mathcal{J}_{f5}$ and $\mathcal{J}_{f6}$, are constant. We split the robot generalised coordinates into $\boldsymbol{q}_f = \left(z_b, \theta_y, \theta_x\right)$ and $\boldsymbol{q}_r = \left(x_b, y_b, \theta_z, \boldsymbol{q}_m\right)$ for which we have: $\boldsymbol{q}_f = \boldsymbol{Q}_f\boldsymbol{q}$ and $\boldsymbol{q}_r = \boldsymbol{Q}_r\boldsymbol{q}$, where $\boldsymbol{Q}_f$ and $\boldsymbol{Q}_r$ are selection matrices. If we left multiply (1) by $\boldsymbol{Q}_r$, and considering that since $\ddot{z}_b = 0$, $\ddot{\theta}_y = 0$ and $\ddot{\theta}_x = 0$ the relation $\ddot{\boldsymbol{q}} = \boldsymbol{Q}_r^T\ddot{\boldsymbol{q}}_r$ holds, we get:

$$\boldsymbol{Q}_r\boldsymbol{B}(\boldsymbol{q})\boldsymbol{Q}_r^T\ddot{\boldsymbol{q}}_r + \boldsymbol{Q}_r\boldsymbol{n}(\boldsymbol{q}, \dot{\boldsymbol{q}}) = \boldsymbol{Q}_r\boldsymbol{S}(\boldsymbol{q})\boldsymbol{u} + \boldsymbol{Q}_r\boldsymbol{A}(\boldsymbol{q})\boldsymbol{\lambda}, \tag{2}$$

while if we left multiply (1) by $\boldsymbol{Q}_f$ we also get:

$$\boldsymbol{Q}_f\boldsymbol{B}(\boldsymbol{q})\boldsymbol{Q}_r^T\ddot{\boldsymbol{q}}_r + \boldsymbol{Q}_f\boldsymbol{n}(\boldsymbol{q}, \dot{\boldsymbol{q}}) = \boldsymbol{Q}_f\boldsymbol{A}(\boldsymbol{q})\boldsymbol{\lambda}. \tag{3}$$

Note that in (3) $\boldsymbol{Q}_f\boldsymbol{S}(\boldsymbol{q})\boldsymbol{u} = \boldsymbol{0}$, since the robot actuators do not perform work on the generalized coordinates $\boldsymbol{q}_f$.

A. State Space reduced Model

Starting from (2) and after some manipulation [19], we can express the robot state space reduced model as:

$$\dot{\boldsymbol{x}} = \boldsymbol{\phi}(\boldsymbol{x}, \boldsymbol{u}) = \begin{pmatrix} \boldsymbol{G}(\boldsymbol{q})\boldsymbol{\nu} \\ \boldsymbol{M}^{-1}(\boldsymbol{q})(\boldsymbol{E}(\boldsymbol{q})\boldsymbol{u} - \boldsymbol{m}(\boldsymbol{q}, \boldsymbol{\nu})) \end{pmatrix}, \tag{4}$$

with the state defined as $\boldsymbol{x} = \left(\boldsymbol{q}_r\,, \boldsymbol{\nu}\right)$, where $\boldsymbol{\nu} = \left(\boldsymbol{\nu}_b\,, \dot{\boldsymbol{q}}_m\right)$ is the robot velocity vector, with $\boldsymbol{\nu}_b$ the base pseudovelocities[1], $\boldsymbol{G}(\boldsymbol{q})$ a matrix whose columns span the null space of $\boldsymbol{A}^T(\boldsymbol{q})\boldsymbol{Q}_r^T$ and

$$\begin{aligned} \boldsymbol{M}(\boldsymbol{q}) &= \boldsymbol{G}^T(\boldsymbol{q})\boldsymbol{Q}_r\boldsymbol{B}(\boldsymbol{q})\boldsymbol{Q}_r^T\boldsymbol{G}(\boldsymbol{q}) \\ \boldsymbol{m}(\boldsymbol{q}, \boldsymbol{\nu}) &= \boldsymbol{G}^T(\boldsymbol{q})\left(\boldsymbol{Q}_r\boldsymbol{B}(\boldsymbol{q})\boldsymbol{Q}_r^T\dot{\boldsymbol{G}}(\boldsymbol{q})\boldsymbol{\nu} + \boldsymbol{Q}_r\boldsymbol{n}(\boldsymbol{q}, \dot{\boldsymbol{q}})\right) \\ \boldsymbol{E}(\boldsymbol{q}) &= \boldsymbol{G}^T(\boldsymbol{q})\boldsymbol{Q}_r\boldsymbol{S}(\boldsymbol{q}). \end{aligned}$$

[1]In the MM used in our simulation, the base pseudovelocities are the driving and steering velocity of the mobile base.

B. Contact Forces

Since we already assumed that the wheel-ground friction is adequate to balance the tangential to the ground forces exerted by the robot, here we will only focus on the elements of $\boldsymbol{\lambda}$ that are orthogonal to the ground. Their determination, however, is not trivial since the balance problem is hyper-static when the contact points are more than 3. Nevertheless, in order to argue about the robot balance, we only need to know the sum of the orthogonal to the ground forces and the moments tangent to the ground that the robot exerts. In (3) the vector $\boldsymbol{Q}_f\boldsymbol{A}(\boldsymbol{q})\boldsymbol{\lambda}$ represents the contact forces at the generalized coordinates level that constrain the base motion along the axes of the joints $\mathcal{J}_{f3}$, $\mathcal{J}_{f5}$ and $\mathcal{J}_{f6}$. If we denote by f_{f3} the force that the robot exerts along the joint $\mathcal{J}_{f3}$ and by μ_{f5} and μ_{f6} the moments that the robot exerts around the joints $\mathcal{J}_{f5}$ and $\mathcal{J}_{f6}$ respectively we have:

$$\begin{pmatrix} f_{f3} \\ \mu_{f5} \\ \mu_{f6} \end{pmatrix} = -\boldsymbol{Q}_f\boldsymbol{A}(\boldsymbol{q})\boldsymbol{\lambda}. \tag{5}$$

Given that $\ddot{\boldsymbol{q}}_r = \boldsymbol{G}(\boldsymbol{q})\dot{\boldsymbol{\nu}} + \dot{\boldsymbol{G}}(\boldsymbol{q})\boldsymbol{\nu}$ and by substituting (3) and (4) in (5), we get f_{f3}, μ_{f5} and μ_{f6} as a function of the robot state and the control inputs:

$$\begin{aligned}\begin{pmatrix} f_{f3} \\ \mu_{f5} \\ \mu_{f6} \end{pmatrix} &= -\boldsymbol{Q}_f\boldsymbol{B}(\boldsymbol{q})\boldsymbol{Q}_r^T\boldsymbol{G}(\boldsymbol{q})\boldsymbol{M}^{-1}(\boldsymbol{q})\boldsymbol{E}(\boldsymbol{q})\boldsymbol{u} \\ &\quad + \boldsymbol{Q}_f\boldsymbol{B}(\boldsymbol{q})\boldsymbol{Q}_r^T\boldsymbol{G}(\boldsymbol{q})\boldsymbol{M}^{-1}(\boldsymbol{q})\boldsymbol{m}(\boldsymbol{q},\boldsymbol{\nu}) \\ &\quad - \boldsymbol{Q}_f\boldsymbol{B}(\boldsymbol{q})\boldsymbol{Q}_r^T\dot{\boldsymbol{G}}(\boldsymbol{q})\boldsymbol{\nu} - \boldsymbol{Q}_f\boldsymbol{n}(\boldsymbol{q},\dot{\boldsymbol{q}}) \end{aligned} \tag{6}$$

Denote by $\mathcal{F}_v$ a frame with axes parallel to $\mathcal{F}_b$ that lies on the projection of $\mathcal{F}_b$ on the ground (see Fig.1). The orthogonal to the ground forces and the tangent to the ground moments that the robot exerts at a point L of the ground can be expressed in $\mathcal{F}_v$ as:

$$\boldsymbol{f}_v = \boldsymbol{R}_w^v(\boldsymbol{q}) f_{f3}\hat{\boldsymbol{z}}_{f3}^w, \tag{7}$$

$$\boldsymbol{\mu}_v = \boldsymbol{R}_w^v(\boldsymbol{q})\left(\mu_{f5}\hat{\boldsymbol{z}}_{f5}^w + \mu_{f6}\hat{\boldsymbol{z}}_{f6}^w + [\boldsymbol{c}]_\times f_{f3}\hat{\boldsymbol{z}}_{f3}^w\right) \tag{8}$$

where $\boldsymbol{R}_w^v(\boldsymbol{q})$ is the rotation matrix of $\mathcal{F}_v$ w.r.t. the $\mathcal{F}_w$, $\hat{\boldsymbol{z}}_{f3}^w$, $\hat{\boldsymbol{z}}_{f5}^w$ and $\hat{\boldsymbol{z}}_{f6}^w$ are the unit vectors of the axes of $\mathcal{J}_{f3}$, $\mathcal{J}_{f5}$ and $\mathcal{J}_{f6}$ respectively, expressed in $\mathcal{F}_w$, $\boldsymbol{c}$ is the position vector that starts from the point of application of $f_{f3}\hat{\boldsymbol{z}}_{f3}^w$ and ends at L expressed in $\mathcal{F}_w$ and $[\boldsymbol{c}]_\times$ is the cross product operator which represents a skew-symmetric matrix built from the elements of $\boldsymbol{c}$.

IV. Proposed NMPC Approach

We will solve this real-time motion generation problem using an appropriate NMPC algorithm. NMPC solves an OCP at each discrete time instant. For an efficient numerical solution, each OCP must be reduced to a Nonlinear Program (NLP). For the solution of the problem, we assume that the robot is always aware of its own state as well as the position and velocity of each obstacle.

Denote by H the prediction horizon, by δ the sampling interval and by $N = H/\delta$ the number of control intervals in the prediction horizon. The decision variables of the NLP to be solved at time t_k are the control inputs $\boldsymbol{U}_k = \{\boldsymbol{u}_{k|0}, ..., \boldsymbol{u}_{k|N-1}\}$ and the robot states $\boldsymbol{X}_k = \{\boldsymbol{x}_{k|0}, ..., \boldsymbol{x}_{k|N}\}$, where $\boldsymbol{u}_{k|i}$ and $\boldsymbol{x}_{k|i}$ are the predicted control inputs and robot state at time t_{k+i} respectively.

Our objective is to make the task error as small as possible while guaranteeing the safety requirements, possibly using the minimum control effort. Denote by $\boldsymbol{y}_{k|i}$ the pose of the end-effector at the predicted time instant t_{k+i}, by $\dot{\boldsymbol{y}}_{k|i}$ the velocity of the end-effector at t_{k+i} for which holds $\dot{\boldsymbol{y}}_{k|i} = \boldsymbol{J}(\boldsymbol{q}_{k|i})\boldsymbol{\nu}_{k|i}$ with $\boldsymbol{J}(\boldsymbol{q}) = \partial\boldsymbol{\sigma}(\boldsymbol{q})/\partial\boldsymbol{q}\boldsymbol{G}(\boldsymbol{q})$ and by $\dot{\boldsymbol{y}}_d(t_{k+i})$ the desired end-effector velocity at t_{k+i}. Denoting the predicted task error at time t_{k+i} as $\boldsymbol{e}_{k|i} = \boldsymbol{y}_d(t_{k+i}) - \boldsymbol{y}_{k|i}$ and its derivative as $\dot{\boldsymbol{e}}_{k|i} = \dot{\boldsymbol{y}}_d(t_{k+i}) - \dot{\boldsymbol{y}}_{k|i}$, we express the running and terminal cost, respectively, as:

$$\begin{aligned} V_{k|i}(\boldsymbol{x}_{k|i}, \boldsymbol{u}_{k|i}) &= \boldsymbol{e}_{k|i}^T\boldsymbol{Q}_p\boldsymbol{e}_{k|i} + \dot{\boldsymbol{e}}_{k|i}^T\boldsymbol{Q}_d\dot{\boldsymbol{e}}_{k|i} \\ &\quad + \boldsymbol{\nu}_{k|i}^T\boldsymbol{P}\boldsymbol{\nu}_{k|i} + \boldsymbol{u}_{k|i}^T\boldsymbol{R}_u\boldsymbol{u}_{k|i} \end{aligned} \tag{9}$$

$$\begin{aligned} V_{k|N}(\boldsymbol{x}_{k|N}) &= \boldsymbol{e}_{k|N}^T\boldsymbol{Q}_{p,N}\boldsymbol{e}_{k|N} + \dot{\boldsymbol{e}}_{k|N}^T\boldsymbol{Q}_{d,N}\dot{\boldsymbol{e}}_{k|N} \\ &\quad + \boldsymbol{\nu}_{k|N}^T\boldsymbol{P}_N\boldsymbol{\nu}_{k|N}. \end{aligned} \tag{10}$$

Here, $\boldsymbol{Q}_p$, $\boldsymbol{Q}_d$ and $\boldsymbol{R}_u$ are the weighting matrices for the task error, its derivative and the control effort throughout the prediction horizon, while $\boldsymbol{Q}_{p,N}$ and $\boldsymbol{Q}_{d,N}$ are the weighting matrices for the task error and its derivative at the final time instant. Note that in order to deal with the robot redundancy, we introduced damping in the robot motion by including the terms $\boldsymbol{\nu}_{k|i}^T\boldsymbol{P}\boldsymbol{\nu}_{k|i}$ and $\boldsymbol{\nu}_{k|N}^T\boldsymbol{P}_N\boldsymbol{\nu}_{k|N}$ in the cost functions, where $\boldsymbol{P}$ and $\boldsymbol{P}_N$ are the weighting matrices for the robot velocity throughout the prediction horizon and at the final time instant respectively.

The NLP that will be solved at time instant t_k is then:

$$\min_{\substack{\boldsymbol{u}_{k|0},...,\boldsymbol{u}_{k|N-1},\\ \boldsymbol{x}_{k|0},...,\boldsymbol{x}_{k|N}}} \sum_{i=0}^{N-1} V_{k|i}(\boldsymbol{x}_{k|i}, \boldsymbol{u}_{k|i}) + V_{k|N}(\boldsymbol{x}_{k|N}) \tag{11a}$$

subject to:

$$\boldsymbol{x}_{k|0} - \boldsymbol{x}_k = 0 \tag{11b}$$

$$\boldsymbol{x}_{k|i+1} - \boldsymbol{\phi}_{\text{d-t}}(\boldsymbol{x}_{k|i}, \boldsymbol{u}_{k|i}) = 0, \quad i = 0, ..., N-1 \tag{11c}$$

$$\boldsymbol{x}_{\min} \le \boldsymbol{x}_{k|i} \le \boldsymbol{x}_{\max}, \quad i = 0, ..., N \tag{11d}$$

$$\boldsymbol{u}_{\min} \le \boldsymbol{u}_{k|i} \le \boldsymbol{u}_{\max}, \quad i = 0, ..., N-1 \tag{11e}$$

$$\textit{collision avoidance constraints at } t_k, ..., t_{k+N} \tag{11f}$$

$$\textit{balance constraints at } t_k, ..., t_{k+N-1} \tag{11g}$$

where $\boldsymbol{x}_k$ represents the current robot state, $\boldsymbol{\phi}_{\text{d-t}}(\cdot,\cdot)$ represents the discrete-time dynamics of the robot obtained via numerical integration under the assumption of piecewise-constant control inputs, while $\boldsymbol{x}_{\min}$, $\boldsymbol{x}_{\max}$ and $\boldsymbol{u}_{\min}$, $\boldsymbol{u}_{\max}$ are respectively the lower/upper bounds on the state variables and on the control inputs.

Regarding the collision avoidance constraints, we use ellipsoids to envelop the n_b robot bodies. Let $\boldsymbol{r}_j(\boldsymbol{x}_{k|i})$ be the position of the center of the ellipsoid that envelops the j-th body at time instant t_{k+i} and α_j, β_j and γ_j be the length of the ellipsoid principal semi-axes. Assuming the frame of the j-th ellipsoid attached at its center with axes aligned with the

ellipsoid principal axes, we denote by $\boldsymbol{R}_j(\boldsymbol{q}_{k|i})$ the rotation matrix of the ellipsoid w.r.t. $\mathcal{F}_w$. To each robot body we associate an obstacle, $\mathcal{O}_j$, that we envelop with a sphere of radius ρ_j, denoting its center by $\boldsymbol{o}_j$. The collision avoidance constraint that is included in (11f) is:

$$(\boldsymbol{r}_j(\boldsymbol{x}_{k|i}) - \boldsymbol{o}_j)^T \boldsymbol{H}_j(\boldsymbol{x}_{k|i})(\boldsymbol{r}_j(\boldsymbol{x}_{k|i}) - \boldsymbol{o}_j) \geq 1 \qquad (12)$$

for $i = 0, ..., N$ and $j = 1, ..., n_b$ with

$$\boldsymbol{H}_j(\boldsymbol{x}_{k|i}) = \boldsymbol{R}_j(\boldsymbol{q}_{k|i}) \boldsymbol{L}_j \boldsymbol{R}_j^T(\boldsymbol{q}_{k|i})$$

and

$$\boldsymbol{L}_j = \text{diag}\{(\alpha_j + \rho_j)^{-2}, (\beta_j + \rho_j)^{-2}, (\gamma_j + \rho_j)^{-2}\}.$$

Note that in the collision avoidance constraints to be included in (11f) we consider also self-collision avoidance constraints for each pair of robot bodies susceptible to collision. The constraints are built similarly to the robot-obstacle collision avoidance constraints assigning appropriately the ellipsoid and the sphere to the two robot bodies based on their shape.

V. Balance Constraint

In this section we first present the criterion that we will use for the evaluation of the robot balance. Then, we derive the constraint to be applied in (11g) and introduce an additional term in the cost function to improve the robot balance.

A. Balance Criterion

As a criterion to evaluate the robot balance, we use the moments that the robot applies around the support polygon edges. The robot maintains its balance if the resulting moments are non-negative. Denote by $\boldsymbol{e}_i$ the unit vector of the i-th edge and by $\boldsymbol{p}_i$ the position of its starting point, both expressed in $\mathcal{F}_v$ (see Fig. 2). If μ_i is the moment that the robot applies around the i-th edge of the support polygon then the criterion for robot balance is expressed as:

$$\mu_i = \boldsymbol{e}_i^T(-\boldsymbol{p}_i' \times \boldsymbol{f}_v + \boldsymbol{\mu}_v) \geq 0, \quad \forall\, i = 1, ..., n_e, \qquad (13)$$

where $\boldsymbol{p}_i' = \boldsymbol{p}_i - \boldsymbol{l}_v$, with $\boldsymbol{l}_v$ the position vector of L w.r.t. $\mathcal{F}_v$ and n_e the number of support polygon edges. Note that constraining the moments around the support polygon edges to remain non-negative is equivalent to constraining the ZMP to lie within the support polygon (see Appendix).

Substituting (7), (8) and (6) in (13) and after simple manipulation we get the balance criterion in the form:

$$\boldsymbol{D}(\boldsymbol{x})\boldsymbol{u} \leq \boldsymbol{b}(\boldsymbol{x}), \qquad (14)$$

where the expressions of $\boldsymbol{D}$ and $\boldsymbol{b}$ are easily derived from the aforementioned equations.

B. Balance Constraint

As we mentioned we will use ACADO for the formulation and solution of the OCP. Although ACADO is very effective in solving an OCP in real-time, it requires a symbolic representation of the OCP. This leads to a memory overhead in cases where the prediction model and the considered constraints are long and complex, as in the case of an MM.

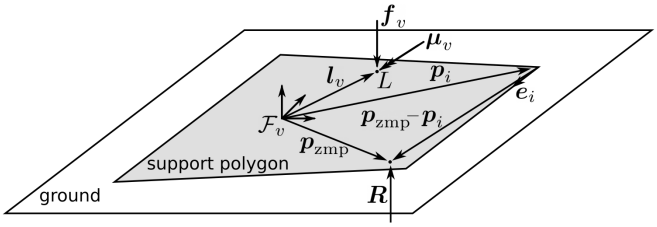

Fig. 2. The robot support polygon. By $\boldsymbol{e}_i$ we denote the unit vector of its i-th edge and by $\boldsymbol{p}_i$ the position of its starting point, all expressed in $\mathcal{F}_v$.

In order to deal with this issue, instead of using the inherently nonlinear balance constraint (14) directly in the NLP, we take advantage of the recursive nature of the NMPC in order to simplify it. Specifically, for the balance constraints of the NLP to be solved at time instant t_k we will use the NLP solution $\boldsymbol{U}_{k-1}$ obtained at time instant t_{k-1}. Starting form the initial state $\boldsymbol{x}_{k-1|0}$ and using the control sequence $\boldsymbol{U}_{k-1}$, we can integrate the robot equations to obtain its trajectory throughout the prediction horizon computed at time instant t_{k-1}, that is $\boldsymbol{X}_{k-1} = \{\boldsymbol{x}_{k-1|0}, \boldsymbol{x}_{k-1|1}, .., \boldsymbol{x}_{k-1|N}\}$. Given that the predictive model is accurate, by applying the control input $\boldsymbol{u}_{k-1|0}$ at time instant t_{k-1} we get the robot initial state at time instant t_k, i.e., $\boldsymbol{x}_{k|0} = \boldsymbol{x}_{k-1|1}$. Assuming no significant changes in the environment between the consecutive time instants t_{k-1} and t_k, we can use $\boldsymbol{X}_{k-1}$ as an estimation of the first $N-1$ states of the trajectory $\boldsymbol{X}_k$. We will use this estimation in order to evaluate $\boldsymbol{D}(\boldsymbol{x})$ and $\boldsymbol{b}(\boldsymbol{x})$ in eq. (14) for each time instant throughout the prediction horizon. The resulting balance constraints that will be applied at (11g) of the NLP to be solved at t_k are now linear combinations of the control inputs:

$$\begin{aligned}
\boldsymbol{D}(\boldsymbol{x}_{k|0})\boldsymbol{u}_{k|0} &\leq \boldsymbol{b}(\boldsymbol{x}_{k|0}) \\
\boldsymbol{D}(\boldsymbol{x}_{k-1|2})\boldsymbol{u}_{k|1} &\leq \boldsymbol{b}(\boldsymbol{x}_{k-1|2}) \\
&\vdots \\
\boldsymbol{D}(\boldsymbol{x}_{k-1|N})\boldsymbol{u}_{k|N-1} &\leq \boldsymbol{b}(\boldsymbol{x}_{k-1|N})
\end{aligned}$$

Note that the solution of the NLP might entail some *predicted* control inputs that lead to unbalanced states, due to the obsolete information used by the previous solution. However, the control input that will be applied to the robot at time instant t_k, namely $\boldsymbol{u}_{k|0}$, guarantees balance at time instant t_k since $\boldsymbol{x}_{k|0}$ is independent of the NMPC solution at t_k.

C. Improving Balance

Even if the balance constraint is satisfied, the robot might get dangerously close to losing balance, unless we introduce an appropriate term in the cost function that improves the robot balance. Thus, we add in the running cost the term $\boldsymbol{v}_{k|i}^T \boldsymbol{\Lambda} \boldsymbol{v}_{k|i}$, where $\boldsymbol{v}_{k|i} = \boldsymbol{D}(\boldsymbol{x}_{k-1|i+1})\boldsymbol{u}_{k|i} - \boldsymbol{b}(\boldsymbol{x}_{k-1|i+1})$ and $\boldsymbol{\Lambda}$ the corresponding weighting matrix. This term helps to evenly distribute the moments among the edges of our support polygon.

VI. SIMULATIONS

In order to show the effectiveness of the proposed method, we conducted a series of simulations assigning to the robot end-effector tasks that require aggressive motions. All the simulations were implemented using Simscape within Simulink on an Intel Core i9-9900K CPU running at 3.60 GHz. For the numerical solution of the NLPs, we used the RTI scheme [17] implemented within ACADO [18].

The robot used for our simulations is an MM consisting of a differential-drive mobile base with two caster wheels carrying a 3-link manipulator on top (see Fig. 3). The height of the mobile base is 0.4 m and the length of the manipulator is 1.35 m. The total robot weight is 44.5 kg. The robot is torque controlled. By τ_r and τ_l we denote the torques at the right and left driving wheels respectively and by τ_1, τ_2 and τ_3 the torques of the actuators at the manipulator joints (the enumeration starts from the closest to the base joint). The MM wheels form a support polygon with four edges. If we locate the frame $\mathcal{F}_v$ along the axis of the robot base, the position of the support polygon edges expressed in this frame is: $\boldsymbol{p}_1 = \big(0.2\,, -0.133\big)$, $\boldsymbol{p}_2 = \big(0.2\,, 0.133\big)$, $\boldsymbol{p}_3 = \big(-0.15\,, 0.2\big)$ and $\boldsymbol{p}_4 = \big(-0.15\,, -0.2\big)$. However, for safety reasons we will considered for the balance constraint a restricted support polygon reducing its dimensions by 10%.

For the collision avoidance constraints, we assign to each robot body its closest obstacle. Their closeness is evaluated using the minimum weighted distance that is considered in (12) between the ellipsoid that envelops the robot body and the sphere that envelops the obstacle. We also include one self-collision avoidance constraint between the base and the third link of the manipulator, enveloping the base in a sphere. The rest of the robot bodies are protected from self-collision thanks to appropriate joint limits.

We wish to compare the proposed method with two approaches from the literature: (1) the NMPC proposed in [16] and (2) the QP proposed in [10]. These methods, originally applied to velocity controlled robots, were slightly modified in order to be applied to the considered torque controlled robot.

In the rest of this section we briefly present the compared methods and then we show the simulation results.

A. Compared Methods

a) NMPC using a ZMP position approximation: The NMPC presented in [16] uses a balance constraint built upon the approximation of the ZMP position:

$$\tilde{\boldsymbol{p}}_{\text{zmp}} = \frac{\boldsymbol{n}_g \times (\boldsymbol{p}_{\text{cog}} \times \boldsymbol{f}_g - \boldsymbol{p}_{ee} \times \boldsymbol{f}_{ee} - \boldsymbol{\mu}_e)}{\boldsymbol{n}_g \cdot (\boldsymbol{f}_g - \boldsymbol{f}_{ee})},$$

where $\boldsymbol{n}_g$ is the ground normal vector, $\boldsymbol{p}_{\text{cog}}$ and $\boldsymbol{f}_g$ are the position of the robot center of gravity and the gravitational forces applied to the robot, $\boldsymbol{p}_{ee}$ the position of the end-effector and $\boldsymbol{f}_{ee}$ and $\boldsymbol{\mu}_e$ are the external forces and moments applied to the end-effector, all expressed in $\mathcal{F}_v$. The considered balance constraint takes the form:

$$g_{\text{zmp}}(\boldsymbol{x}) = \rho_{sc}^2 - \|\tilde{\boldsymbol{p}}_{\text{zmp}}\|^2 \geq 0 \tag{15}$$

where ρ_{sc} is the radius of a circle enveloped by the support polygon. In [16], the balance constraint (15) was included in the cost function using a relaxed barrier function. However, for the purpose of this comparison we will use an NLP as the one proposed in (11) using (15) as balance constraint. To improve the robot balance we include in the running and terminal costs (9) and (10) the terms $w(\ln\left(g_{\text{zmp}}(\boldsymbol{x}_{k|i})/\rho_{sc}^2\right))^2$ and $w_N(\ln\left(g_{\text{zmp}}(\boldsymbol{x}_{k|N})/\rho_{sc}^2\right))^2$ where w and w_N are the associated weights. In the simulation results we will refer to this method as NMPC-CG (CG stands for center of gravity).

b) QP using the STC: The method in [10] uses a QP in order to solve the inverse kinematics problem given an end-effector task at the velocity level. For the purpose of this comparison, we slightly modified the QP in order to consider the robot dynamics. If $\boldsymbol{q}_{k+i}$, $\boldsymbol{\nu}_{k+i}$, $\boldsymbol{x}_{k+i}$ and $\boldsymbol{u}_{k+i}$ are the robot configuration, velocity, state and control inputs at time t_{k+i} respectively, then the QP to be solved at t_k is:

$$\begin{aligned}
&\min_{\boldsymbol{u}_k} \ \boldsymbol{\nu}_{k+1}^T \boldsymbol{Q} \boldsymbol{\nu}_{k+1} + \boldsymbol{v}_k^T \boldsymbol{P}_v \boldsymbol{v}_k + \boldsymbol{u}_k^T \boldsymbol{R}_u \boldsymbol{u}_k \\
&\text{subject to:} \\
&\dot{\boldsymbol{y}}_d(t_{k+1}) - \boldsymbol{J}(\boldsymbol{q}_{k+1})\boldsymbol{\nu}_{k+1} = \boldsymbol{0} \\
&\boldsymbol{q}_{\min} \leq \boldsymbol{q}_{k+2} \leq \boldsymbol{q}_{\max} \\
&\boldsymbol{\nu}_{\min} \leq \boldsymbol{\nu}_{k+1} \leq \boldsymbol{\nu}_{\max} \\
&\boldsymbol{u}_{\min} \leq \boldsymbol{u}_k \leq \boldsymbol{u}_{\max} \\
&\boldsymbol{D}(\boldsymbol{x}_k)\boldsymbol{u}_k \leq \boldsymbol{b}(\boldsymbol{x}_k)
\end{aligned}$$

where $\boldsymbol{q}_{k+1}$, $\boldsymbol{q}_{k+2}$ and $\boldsymbol{\nu}_{k+1}$ result from the integration of the equations of motion (4) using the Euler method and they are functions of $\boldsymbol{x}_k$ and $\boldsymbol{u}_k$. The balance is maintained using (14) while the term $\boldsymbol{v}_k^T \boldsymbol{P}_v \boldsymbol{v}_k$ improves the robot balance with $\boldsymbol{v}_k = \boldsymbol{D}(\boldsymbol{x}_k)\boldsymbol{u}_k - \boldsymbol{b}(\boldsymbol{x}_k)$ and $\boldsymbol{P}_v$ the associated weight. Note that here we cannot apply collision avoidance constraints since in principle they are nonlinear functions of the state, while the QP can only accept constraints formulated as linear combinations of the control inputs. We will refer to this method as QP-STC and for its solution we use `quadprog`.

B. Simulation Results

We compare the three methods in five different scenarios of assigned end-effector tasks. For all methods the sampling interval is $\delta = 23$ ms while the prediction horizon for the proposed method and the NMPC-CG is $H = 0.23$ s, chosen as large as possible to allow real-time performance. In Fig. 3 we offer a series of snapshot of the simulations while at `https://youtu.be/xp3qVcyYww8` we offer the full video of the simulations. Note that the maximum iteration time for the proposed method was less than 23 ms in all considered scenarios, showing that it can perform in real-time.

Scenario 1: The assigned end-effector task is to track a linear path of length 1.45 m following a trapezoidal velocity profile. The task duration is 4 s and the maximum acceleration along the path is $0.44\,\text{m/s}^2$. The environment is obstacle-free. In this scenario all three methods generate feasible motions (see Fig. 3 and video). However, Fig. 4 shows that the proposed method and the NMPC-CG (inseparable in the plot) give a smoother ZMP path than the QP-STC.

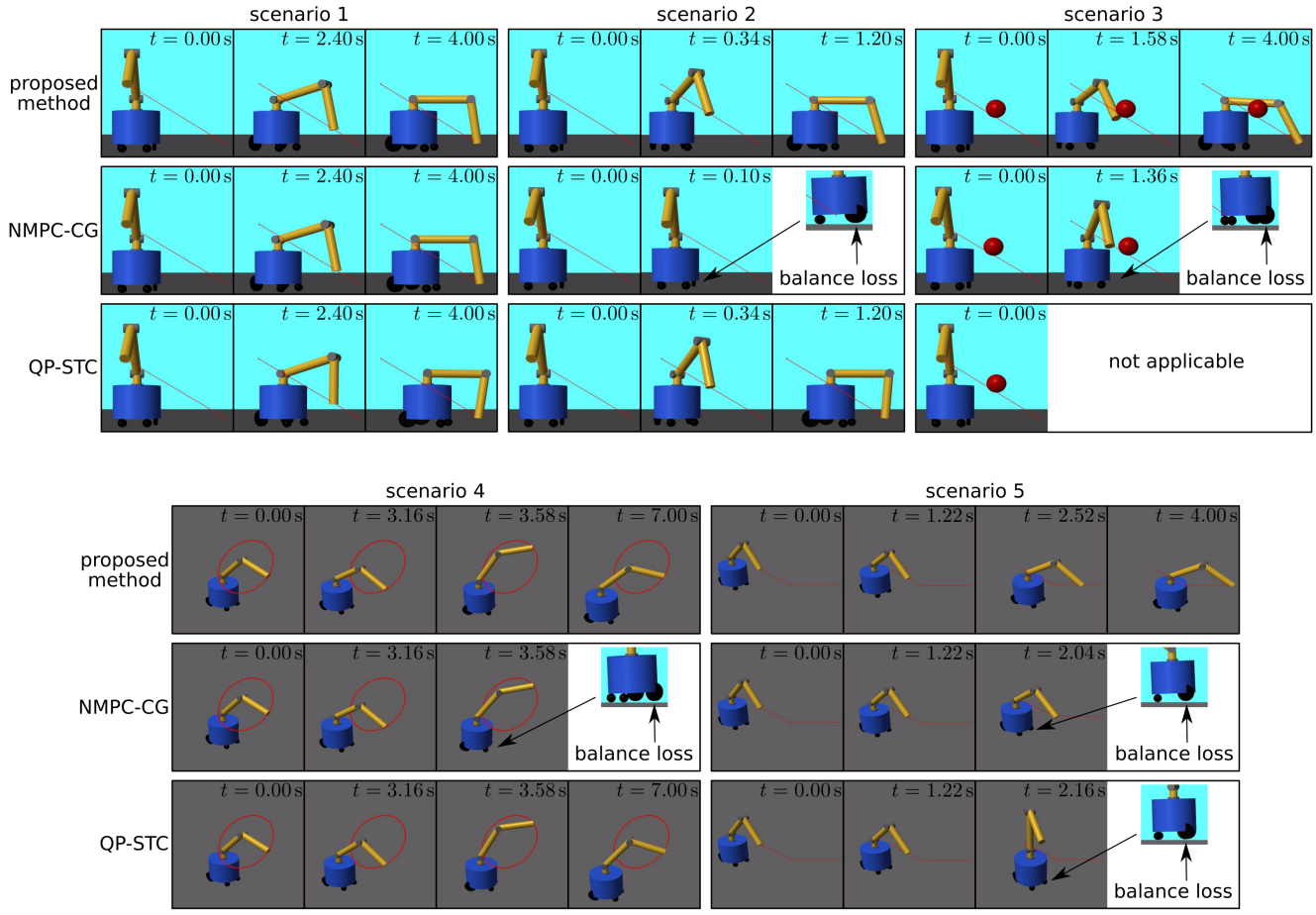

Fig. 3. Snapshots of the motion generated by the proposed method and the two compared methods in the five scenarios.

Scenario 2: The assigned end-effector task is the same of Scenario 1 only now the task duration is decreased to $1.2\,\mathrm{s}$ while the maximum acceleration is increased to $4.84\,\mathrm{m/s^2}$. The environment is again obstacle-free. Fig. 3 (and the video) shows that only the methods that consider the full robot dynamics for the evaluation of the robot balance, i.e., the proposed method and QP-STC, are able to generate feasible motions while the NMPC-CG fails as the rear wheels of the MM lose contact with the ground. It is apparent that, unlike Scenario 1, here the high acceleration required at the beginning of the robot motion makes the ZMP position approximation of the NMPC-CG inadequate and the robot loses balance immediately (see Fig. 5 for the ZMP position).

Scenario 3: The assigned end-effector task is the same of Scenario 1 but now the desired path is obstructed by a spherical obstacle of radius $0.1\,\mathrm{m}$ (see Fig. 3). The presence of the obstacle leads to an increase of the required acceleration at its vicinity (see Fig. 7). In Fig. 3 (and the video) we can see that while our method performs satisfactory, the NMPC-CG fails and the robot loses its balance when it is called to avoid the obstacle (see Fig. 6). Note that as we mentioned in Section VI-A, the QP-STC cannot be applied in this scenario since it does not support collision avoidance constraints.

Scenario 4: The end-effector has to follow a circular path completing two full circles in $7\,\mathrm{sec}$. The radius of the circle is $0.5\,\mathrm{m}$. The environment is obstacle-free. In this scenario, only the proposed method and the QP-STC were able to complete the task, while in Fig. 3 (and the video) we can see the NMPC-CG losing balance as the rear wheels of the robot detach from the ground at the beginning of the second circle (see also Fig. 8 for the ZMP position). In Fig. 8 we can also see that the proposed method generates motions that result to a smoother ZMP path than the QP-STC.

Scenario 5: The end-effector has to follow a linear path of length $0.7\,\mathrm{m}$ followed by a second linear path of length $0.7\,\mathrm{m}$ that forms an angle of $\pi/4\,\mathrm{rad}$ with the first one (see Fig. 3). The desired velocity has again a trapezoidal profile. The task duration is $4\,\mathrm{s}$ and the maximum acceleration along the path is $0.44\,\mathrm{m/s^2}$. The environment is obstacle-free. In Fig. 3 (and the video) we can see that due to the abrupt change of direction, that leads to increased required accelerations (see Fig. 10), only the proposed method is able to generate feasible motions, while the rest lose balance (see also Fig. 9).

VII. Conclusion

In this work we presented a novel real-time motion generation approach for MMs using an NMPC. Kinodynamic feasibility is enforced through the use of the robot dynamic

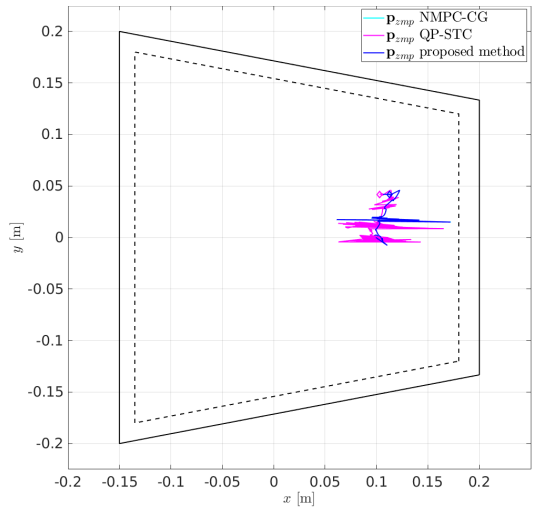

Fig. 4. The ZMP position resulting in Scenario 1

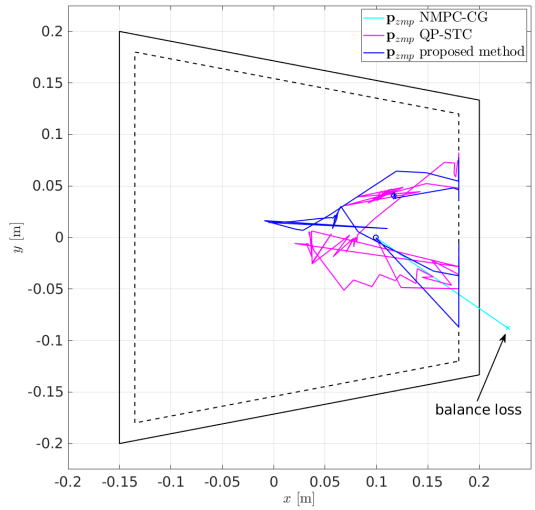

Fig. 5. The ZMP position resulting in Scenario 2

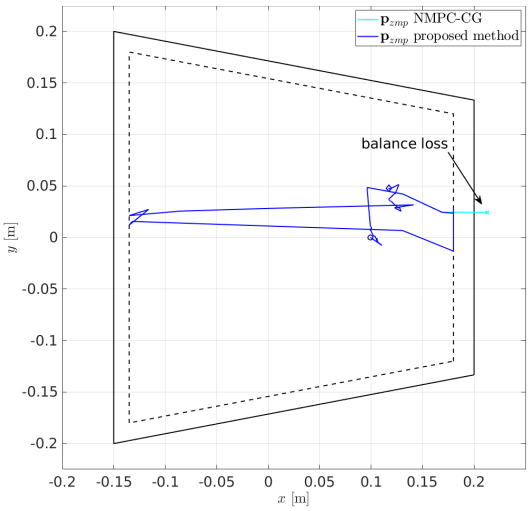

Fig. 6. The ZMP position resulting in Scenario 3

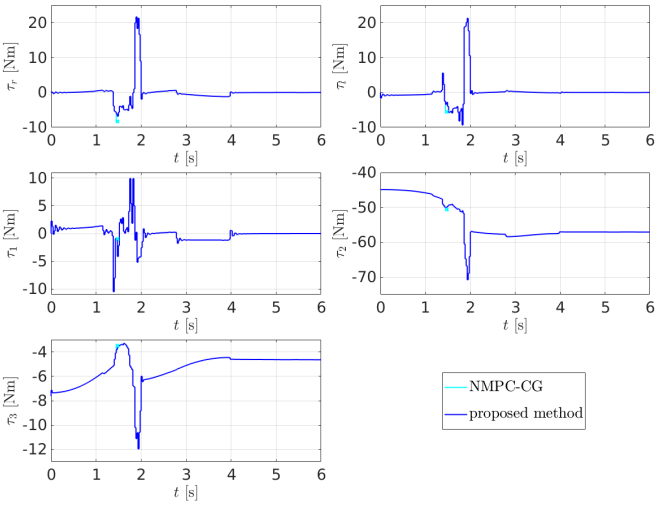

Fig. 7. Control inputs resulting in Scenario 3

model as prediction model and via input and state constraints. To prevent the robot loss of balance in cases where it is called to execute aggressive motions we included a constraint that restricts the robot feasible motions to those that result to non-negative moments around the support polygon edges. To enable the solution of the proposed NMPC scheme with off-the-self solvers that require symbolic representation of the OCP, like ACADO, we lifted the inherent nonlinearity of the balance constraint by linearizing it using the solution of the previous iteration of the NMPC. The proposed method was compared with two other methods in five scenarios. The simulation results showed that the proposed method can effectively handle end-effector tasks that require aggressive motions even in cases where the other methods fail.

Future work aims at the experimental validation of the proposed method as well as the extension of the proposed method to consider additional aspects of the robot dynamics (e.g., the effect of the wheel-ground friction).

APPENDIX

Using Fig. 2 as a reference, we will show the relation between the moment around the i-th support polygon edge, $\boldsymbol{\mu}_i$ and ZMP position, $\boldsymbol{p}_{\text{zmp}}$. We can obtain the ZMP position from the static equilibrium:

$$\boldsymbol{R} + \boldsymbol{f}_v = \boldsymbol{0} \tag{16a}$$

$$\boldsymbol{p}_{\text{zmp}} \times \boldsymbol{R} + \boldsymbol{l}_v \times \boldsymbol{f}_v + \boldsymbol{\mu}_v = \boldsymbol{0} \tag{16b}$$

where $\boldsymbol{R}$ the ground support force due to the contact of the robot with the ground. Substituting (16a) and $\boldsymbol{l}_v = \boldsymbol{p}_i - \boldsymbol{p}_i'$ in (16b) we get:

$$-\boldsymbol{p}_i' \times \boldsymbol{f}_v + \boldsymbol{\mu}_v = \boldsymbol{p}_{\text{zmp}} \times \boldsymbol{f}_v - \boldsymbol{p}_i \times \boldsymbol{f}_v \tag{17}$$

Substituting (17) in (13) we get the relation between the moment around the i-th edge of the support polygon and the position of the ZMP, that is:

$$\boldsymbol{\mu}_i = \boldsymbol{f}_v^T (\boldsymbol{e}_i \times (\boldsymbol{p}_{\text{zmp}} - \boldsymbol{p}_i)) \tag{18}$$

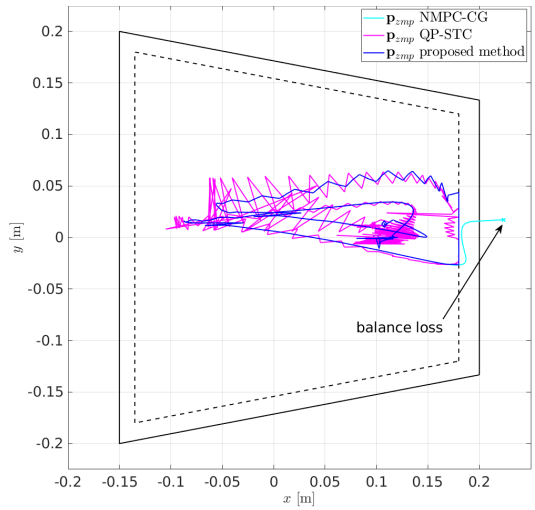

Fig. 8. The ZMP position resulting in Scenario 4

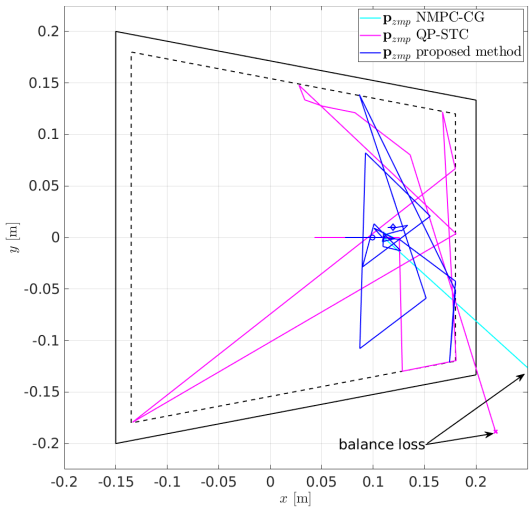

Fig. 9. The ZMP position resulting in Scenario 5

From this relation we can deduce that:

- if the ZMP lies on the half-plane that is defined by e_i and contains the support polygon, then $\mu_i > 0$;
- if the ZMP lies on the half-plane that is defined by e_i but it does not contain the support polygon, then $\mu_i < 0$;
- if the ZMP lies on the line generated by $\boldsymbol{e}_i$, then $\mu_i = 0$.

REFERENCES

[1] S. Dubowsky and E. E. Vance, "Planning mobile manipulator motions considering vehicle dynamic stability constraints," in *1989 IEEE Int. Conf. on Robotics and Automation*, 1989, pp. 1271–1276.

[2] E. G. Papadopoulos and D. A. Rey, "A new measure of tipover stability margin for mobile manipulators," in *1996 IEEE Int. Conf. on Robotics and Automation*, 1996, pp. 3111–3116.

[3] D. A. Rey and E. G. Papadoupoulos, "Online automatic tipover prevention for mobile manipulators," in *in 1997 IEEE/RSJ Int. Conf. on Intelligent Robot and Systems*, 1997, pp. 1273–1278.

[4] M. Vukobratovic and B. Borovac, "Zero-moment point - thirty five years of its life." *The Int. J. of Humanoid Robotics*, vol. 1, pp. 157–173, 2004.

[5] S. Sugano, Q. Huang, and I. Kato, "Stability criteria in controlling mobile robotic systems," in *1993 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, 1993, pp. 832–838.

[6] Q. Huang, S. Sugano, and I. Kato, "Stability control for a mobile manipulator using a potential method," in *1994 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, 1994, pp. 839–846.

[7] Q. Huang, K. Tanie, and S. Sugano, "Coordinated motion planning for a mobile manipulator considering stability and manipulation." *The Int. J. of Robotic Research*, vol. 19, pp. 732–742, 2000.

[8] J. Kim, W. K. Chung, Y. Youm, and B. H. Lee, "Real-time ZMP compensation method using null motion for mobile manipulators," in *2002 IEEE Int. Conf. on Robotics and Automation*, 2002, pp. 1967–1972.

[9] S. Lee, M. Leibold, M. Buss, and F. C. Park, "Rollover prevention of mobile manipulators using invariance control and recursive analytic zmp gradients," *Advanced Robotics*, vol. 26, no. 11-12, pp. 1317–1341, 2012.

[10] C. Qiu, Q. Cao, L. Yu, and S. Miao, "Improving the stability level for on-line planning of mobile manipulators," *Robotica*, vol. 27, no. 3, p. 389–402, 2009.

[11] L. Yu, Q. Cao, C. Li, and C. Qiu, "On-line planning of nonholonomic mobile manipulators based on stability twist constraint," *Journal of the Brazilian Society of Mechanical Sciences and Engineering*, vol. 32, pp. 165–170, 2010.

[12] X. Ding, Y. Liu, J. Hou, and Q. Ma, "Online dynamic tip-over avoidance for a wheeled mobile manipulator with an improved tip-over moment stability criterion," *IEEE Access*, vol. 7, pp. 67 632–67 645, 2019.

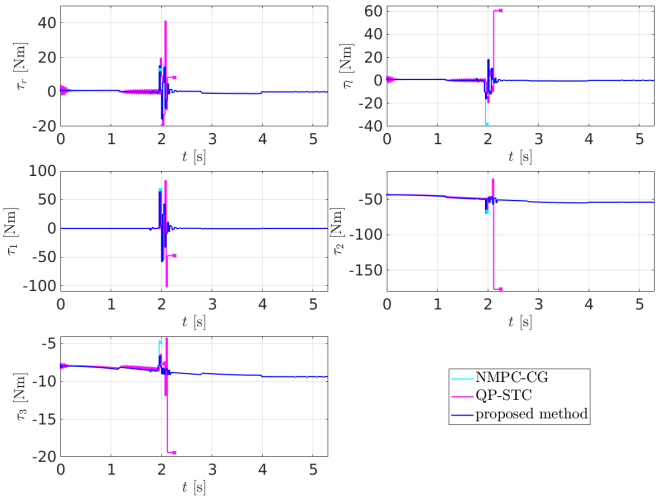

Fig. 10. Control inputs resulting in Scenario 5

[13] G. Buizza Avanzini, A. M. Zanchettin, and P. Rocco, "Constrained model predictive control for mobile robotic manipulators," *Robotica*, vol. 36, no. 1, p. 19–38, 2018.

[14] M. Giftthaler, F. Farshidian, T. Sandy, L. Stadelmann, and J. Buchli, "Efficient kinematic planning for mobile manipulators with non-holonomic constraints using optimal control," in *2017 IEEE Int. Conf. on Robotics and Automation*, 2017, pp. 3411–3417.

[15] A. Sideris and J. Bobrow, "An efficient sequential linear quadratic algorithm for solving nonlinear optimal control problems," in *2005 American Control Conf.*, 2005, pp. 2275–2280 vol. 4.

[16] J. Pankert and M. Hutter, "Perceptive model predictive control for continuous mobile manipulation," *IEEE Robotics and Automation Letters*, vol. 5, no. 4, pp. 6177–6184, 2020.

[17] M. Diehl, H. Bock, J. P. Schlöder, R. Findeisen, Z. Nagy, and F. Allgöwer, "Real-time optimization and nonlinear model predictive control of processes governed by differential-algebraic equations," *Journal of Process Control*, vol. 12, no. 4, pp. 577–585, 2002.

[18] B. Houska, H. J. Ferreau, and M. Diehl, "An auto-generated real-time iteration algorithm for nonlinear mpc in the microsecond range," *Automatica*, vol. 47, no. 10, pp. 2279 – 2285, 2011.

[19] B. Siciliano, L. Sciavicco, L. Villani, and G. Oriolo, *Robotics: Modelling, Planning and Control*. Springer, 2008.