Robotica (2018) volume 36, pp. 19–38. © Cambridge University Press 2017
doi:10.1017/S0263574717000133

Constrained model predictive control for mobile robotic manipulators

Giovanni Buizza Avanzini*, Andrea Maria Zanchettin and Paolo Rocco

Politecnico di Milano, Dipartimento di Elettronica, Informazione e Bioingegneria, Piazza L. Da Vinci 32, 20133, Milano, Italy. E-mails: andreamaria.zanchettin@polimi.it, paolo.rocco@polimi.it

(Accepted March 8, 2017. First published online: April 3, 2017)

SUMMARY
This paper discusses the application of a constraint-based model predictive control (MPC) to mobile manipulation tracking problems. The problem has been formulated so as to guarantee offset-free tracking of piecewise constant references, with convergence and recursive feasibility guarantees. Since MPC inputs are recomputed at every control iteration, it is possible to deal with dynamic and unknown scenarios. A number of motion constraints can also be easily included: Acceleration, velocity and position constraints have been enforced, together with collision avoidance constraints for the mobile base and the arm and field-of-view constraints. Such constraints have been extended over the prediction horizon maintaining a linear-quadratic formulation of the problem. Navigation performance has been improved by devising an online algorithm that includes an additional goal to the problem, derived from the classical vortex field approach. Experimental validation shows the applicability of the proposed approach.

KEYWORDS: Control of robotic systems; Mobile manipulators; Predictive control; Redundant manipulators; Navigation; Motion planning.

1. Introduction

Online-solvable instantaneous constrained optimization is getting increasingly popular in recent years in the context of robot control (e.g., refs. [1–3]), thanks to the easy integration of kinematics, velocity and acceleration constraints into the problem formulation, without the need of pre-planned trajectories. Two approaches are mainly used to define the robot task for constrained optimization. One is to use the task formalism,[4] that treats the robot task as a constraint to be respected while minimizing a suited cost function (i.e., the optimization problem cost function). Conversely, the task error itself can be considered as the (quadratic) cost function of the optimization problem to be minimized, while constraints are used to specify additional requirements to be fulfilled by the problem solution, usually as state and/or input limitations.

One of the problems of instantaneous optimization is however its "blindness"[5] with respect to the system evolution. In fact, the future state and constraints behavior are not included in the problem definition, so that a feasible solution at a given time instant can lead to an unfeasible configuration in the future. Optimal control, on the other hand, takes into account the system evolution, and possibly also the environment one, if an appropriate model is available, but it does not allow to include constraints on the state and input variables. Model predictive control or MPC[6,7] instead allows the implementation of a state-feedback controller (through the Receding Horizon Principle) that is optimal over the future N time steps (the so called prediction horizon), and also complies to possibly enforced constraints on the state and input variables. The solution of the MPC problem is the control input sequence that minimizes the problem cost function while complying with the imposed constraints. The use of MPC for mechatronic system control has not been very common, mainly due

* Corresponding author. E-mail: giovannimassimo.buizza@polimi.it

to the high computational cost that is usually involved compared to the high frequency sampling rates typical of such systems. However, recent advancements in computational performance, and the use of tailored solution algorithms,[8] can now allow the implementation of MPC also for on-line control of fast dynamic systems such as robots.

In this context, we are interested in the control of omnidirectional mobile manipulators, moving in a dynamic environment while subject to a number of constraints (acceleration, velocity and position constraints for all the joints, collision avoidance constraints both for the arm and the mobile base, field-of-view (FoV) constraints), and possibly cooperating with humans and interacting with the environment. With "mobile manipulator," we intend a robot made up of a mobile robot base and of a robotic arm mounted on it. The practically unlimited workspace and the usual high degree of redundancy of mobile manipulators make them particularly interesting for both industrial and domestic use.

The use of MPC for *mobile robots* navigation is currently investigated in the literature. Among the others, in ref. [9] the authors use a dipolar vector field and non-linear MPC with FoV constraints to navigate a non-holonomic vehicle in presence of obstacles, presenting simulation results. In ref. [10], non-linear MPC (NMPC) is used to control a formation of omnidirectional mobile robots for target tracking, presenting experimental validation. Each robot solves its own MPC problem, sharing information about its planned trajectory with the other robots and dynamically adapting the MPC parameters. This approach is further developed in ref. [11], where the NMPC is combined with an Artificial Potential Field to also obtain obstacle avoidance, while switching to the more computationally intensive A^* algorithm when dealing with local minima problems. In ref. [12], Rapidly-exploring Random Tree (RRT) is combined with MPC for mobile robot navigation, imposing kinodynamics constraints on the vehicle and validating the proposed algorithm in a real-world scenario. Linear MPC is used in ref. [13] to control an omnidirectional mobile robot while compensating for static friction, with experimental validation.

On the other hand, in the *mobile manipulation* field, MPC has been less investigated, and different control approaches have been proposed. A unified framework for feedback control of both holonomic and non-holonomic mobile manipulators is introduced in ref. [14], extending the transverse function approach developed for mobile robots while guaranteeing joint limits compliance. Obstacle avoidance is however not included in this approach and only simulation results are proposed. In ref. [15], a framework for the kinematic and dynamic modeling of wheeled mobile manipulators is provided, in addition to a reactive control scheme that uses potential function to enforce a number of constraints on the robot motion (joint limits, rated actuator inputs, singularities and obstacle avoidance). Simulated and experimental results are presented, though no actual obstacles perception (and thus reaction) is performed in the experimental validation. In ref. [16], a low-level flexible planning method for mobile manipulators is presented and experimentally tested. Distance information provided by distance sensors is used to modify online pre-planned motions path in order to avoid collisions with perceived obstacles, while exploiting redundancy to maintain consistency with the desired path if it is possible. In ref. [17], a mobile manipulator is controlled through MPC for visual servoing tasks. The image features are maintained in the camera FoV by suitable state constrains. The Visual Tracking Algorithm is triggered only when it is necessary to ensure the convergence of the robot to the desired position, thus reducing the computational load. However, no manipulation task is actually performed by the robot, and no obstacles are present while the robot is moving. In ref. [18], MPC is actually used for controlling a mobile manipulator, presenting experimental results of real-time application on a platform composed of a mobile base and of a three degrees of freedom (DoF) manipulator. However, obstacles are not considered by the authors, and only state and input constraints are enforced.

In this paper, we present an online MPC strategy for omnidirectional mobile manipulators. The robot has to move its end-effector toward (possibly time varying) desired configurations, navigating in an unknown environment. The goal is to safely cooperate with humans, while complying with a number of constraints (acceleration, velocity and position constraints for all the joints, collision avoidance constraints, both for the arm and the mobile base, and FoV constraints). Preliminary results were presented in refs. [19, 20]. In particular, in this paper the following advancements are discussed. First, in order to solve the problem, a MPC problem in the velocity form[21] is formulated instead of the classical tracking formulation used in ref. [20]. Combining the velocity form formulation with the results of ref. [22] allows offset-free tracking of piecewise constant reference signals, with recursive feasibility and asymptotic stability guarantees.[23] To our knowledge, this formulation has never been used for online robot control. An artificial safe vortex field is included in the MPC formulation to help reactive autonomous navigation. As a second contribution, a specific algorithm that computes such vortex field exploiting only sensor-based information about the environment is here presented. As a third contribution, an inverse kinematics procedure is proposed for omnidirectional mobile manipulators endowed with an anthropomorphic arm. The whole approach has been experimentally validated on a mobile manipulator,[24] endowed with depth, distance, and vision sensors for environment and human perception. With respect to ref. [20], all of the obstacles are perceived through exteroceptive sensors, without requiring any a-priori information. The MPC problem is solved online at each control cycle, providing the input references for the robot base and the arm, treated as a single system. It does not require any offline pre-planned path.

The remainder of this work is organized as follows: Section 2 describes the MPC problem formulation in velocity form for tracking joint space references. Section 3 presents the definition of an artificial vortex field for sensor-based, online navigation, including the field gradient in the MPC cost function. Section 4 deals with tracking Cartesian space reference, also presenting a robot-independent redundancy resolution strategy. In Section 5, the enforced constraints are formulated and extended over the MPC prediction horizon. Finally, Section 6 presents the experimental setup and results, while Section 7 summarizes the paper contributions.

2. MPC for Tracking in the Velocity Form

Let us consider a n DoF mobile manipulator, made up of a 3 DoF omnidirectional mobile base and of a $n_a = n - 3$ DoF robotic arm. Let us also assume that the robot is velocity controlled, as it is common. In this paper, we will concentrate on high-level trajectory planning level, assuming that the low-level controllers can adequately track references at the velocity level. However, constraints on the maximum acceleration will be included in order to avoid excessive stress on the actuators and on the mechanical system. With these premises, the purely kinematic model of the system is reduced to n decoupled integrators. Therefore, the discrete-time joint-space model of the robot at time instant k takes the form:

$$\mathbf{q}_{k+1} = \mathbf{A}\mathbf{q}_k + \mathbf{B}\mathbf{u}_k, \tag{1}$$

where $\mathbf{q} = [\mathbf{q}_b^{\mathrm{T}}, \mathbf{q}_a^{\mathrm{T}}]^{\mathrm{T}} \in \mathbb{R}^n$ is the robot state vector, containing the base and arm joint variables. $\mathbf{q}_b = [x, y, \theta]_b^{\mathrm{T}} \in \mathbb{R}^3$ refers to the base joint coordinates (planar positions and orientation), and $\mathbf{q}_a = [q_1, \ldots, q_{n_a}]^{\mathrm{T}} \in \mathbb{R}^{n_a}$ to the arm joints. $\mathbf{A} = \mathbf{I}_n \in \mathbb{R}^{n\times n}$ is the (identity) state matrix, $\mathbf{B} = \Delta T \mathbf{I}_n \in \mathbb{R}^{n\times n}$ is the input matrix, with ΔT being the discrete time interval. The input state vector $\mathbf{u} = \dot{\mathbf{q}} = \mathrm{d}\mathbf{q}/\mathrm{d}t \in \mathbb{R}^n$ is the robot reference velocity. Assume the state accessible and note that $(\mathbf{A}, \mathbf{B})$ is reachable.

Suppose that at time instant k the robot task is to reach a certain configuration $\mathbf{q}_k^d$ while complying with a number of motion constraints, namely $\mathbf{q} \in \mathbb{Q}$ and $\mathbf{u} \in \mathbb{U}$, where $\mathbb{Q}$ and $\mathbb{U}$ are compact convex polytopes containing the origin, i.e.,

$$\begin{aligned} \mathbb{Q} &= \{\mathbf{q} \in \mathbb{R}^n : \mathbf{G}_q\,\mathbf{q} \le \mathbf{b}_q\} \\ \mathbb{U} &= \{\mathbf{u} \in \mathbb{R}^n : \mathbf{G}_u\,\mathbf{u} \le \mathbf{b}_u\} \end{aligned} \tag{2}$$

(see Section 5 for details on the enforced constraints). The desired goal configuration $\mathbf{q}_k^d$ is allowed to change in time, resulting in a piecewise constant reference signal. The MPC approach in the tracking form can then be effectively used to obtain control inputs that comply with the constraints, while moving the state toward the desired configuration. Thanks to the receding horizon (RH) principle, MPC can in fact deal with time varying scenarios, and constraints formulated linearly with respect to the input $\mathbf{u}$ can be easily included in the problem.

When using MPC for tracking, it is wise to include an integral action in the control loop to avoid offsets in presence of disturbances or model errors. The velocity form formulation can be used to implicitly include such integrators in the receding horizon control loop.[21] The velocity form is based on redefining the MPC tracking problem in terms of state and control *increments*:

$$\begin{cases} \boldsymbol{\delta}\mathbf{q}_k = \mathbf{q}_k - \mathbf{q}_{k-1} \in \mathbb{R}^n & \text{state increment} \\ \boldsymbol{\delta}\mathbf{u}_k = \mathbf{u}_k - \mathbf{u}_{k-1} \in \mathbb{R}^n & \text{control increment.} \end{cases} \tag{3}$$

Besides the use of the velocity form, some considerations on convergence and recursive feasibility must be done when tracking a piecewise constant reference Notice that an MPC problem is called recursively feasible if, for all feasible initial states, feasibility is guaranteed at every state along the closed-loop trajectory. As it is common in MPC, in fact, a terminal weight and a terminal constraint should be used to guarantee convergence to the desired set-point.[7] However, if a modification occurs in the desired set-point, the MPC can no longer be feasible, i.e., the set-point can correspond to steady-state and input values that are outside the terminal constraint set, or outside the constraint set of Eq. (2). To solve this issue, an additional *artificial* reference $\bar{\mathbf{q}}_k$ can then be introduced:[25] $\bar{\mathbf{q}}_k$ will be considered in the MPC problem as an additional optimization variable, that will be constrained to be always feasible, and it will be used as reference for the state vector $\mathbf{q}$, instead of the actual set point $\mathbf{q}_k^d$ (left unconstrained). The error between $\bar{\mathbf{q}}_k$ and $\mathbf{q}_k^d$ will then be penalized in the cost function, in order to guarantee that the state actually evolves toward the actual set point $\mathbf{q}_k^d$ (see refs. [22, 25–27] for details on the artificial set point, and ref. [23] for an integration of this approach with the velocity form, and for recursive feasibility and stability guarantees).

To include the artificial set point, we should define

$$\begin{cases} \mathbf{e}_k = \mathbf{q}_k - \bar{\mathbf{q}}_k \ \in \mathbb{R}^n & \text{set-point error} \\ \boldsymbol{\xi} = [\boldsymbol{\delta}\mathbf{q}^{\mathrm{T}},\ \mathbf{e}^{\mathrm{T}}]^{\mathrm{T}} \in \mathbb{R}^{2n} & \text{augmented state} \\ \boldsymbol{\zeta}_k = [\boldsymbol{\xi}_k^{\mathrm{T}},\ \bar{\mathbf{q}}_k^{\mathrm{T}}]^{\mathrm{T}} \in \mathbf{R}^{3n} & \text{“terminal” augmented state.} \end{cases} \tag{4}$$

Combining Eqs. (1), (3) and (4), we obtain the augmented system dynamic equation:

$$\boldsymbol{\xi}_{k+1} = \begin{bmatrix} \mathbf{A} & \mathbf{0} \\ \mathbf{A} & \mathbf{I}_n \end{bmatrix} \boldsymbol{\xi}_k + \begin{bmatrix} \mathbf{B} \\ \mathbf{B} \end{bmatrix} \boldsymbol{\delta}\mathbf{u}_k = \mathbf{A}_v \boldsymbol{\xi}_k + \mathbf{B}_v \boldsymbol{\delta}\mathbf{u}_k. \tag{5}$$

Let us also define the sequence of the future control variable increments on the prediction horizon, composed of N time steps, as $\boldsymbol{\delta}\mathbf{U}_k = [\boldsymbol{\delta}\mathbf{u}_k^{\mathrm{T}}, \boldsymbol{\delta}\mathbf{u}_{k+1}^{\mathrm{T}}, \ldots, \boldsymbol{\delta}\mathbf{u}_{k+N-1}^{\mathrm{T}}]^{\mathrm{T}}$. The MPC cost function for the tracking problem in the velocity form is then defined as

$$L(\boldsymbol{\delta}\mathbf{U}_k, \bar{\mathbf{q}}_k; \boldsymbol{\delta}\mathbf{q}_k, \mathbf{q}_k^d) = \sum_{i=0}^{N-1} \left(\|\boldsymbol{\xi}_{k+i}\|_{\mathbf{Q}}^2 + \|\boldsymbol{\delta}\mathbf{u}_{k+i}\|_{\mathbf{R}}^2 \right) + \|\boldsymbol{\xi}_{k+N}\|_{\mathbf{P}}^2 + \|\bar{\mathbf{q}}_k - \mathbf{q}_k^d\|_{\mathbf{T}}^2, \tag{6}$$

where $\mathbf{Q} \in \mathbb{R}^{2n\times 2n}$, $\mathbf{R} \in \mathbb{R}^{n\times n}$, $\mathbf{P} \in \mathbb{R}^{2n\times 2n}$ and $\mathbf{T} \in \mathbb{R}^{n\times n}$ are all positive definite weight matrices. The MPC problem for tracking in the velocity form is then, $\forall i = 0, \ldots, N-1$:

$$\begin{array}{ll} \min\limits_{\boldsymbol{\delta}\mathbf{U}_k, \bar{\mathbf{q}}_k} & L(\boldsymbol{\delta}\mathbf{U}_k, \bar{\mathbf{q}}_k; \boldsymbol{\delta}\mathbf{q}_k, \mathbf{q}_k^d) \\ \text{s.t.} & \boldsymbol{\xi}_{k+i+1} = \mathbf{A}_v\, \boldsymbol{\xi}_{k+i} + \mathbf{B}_v\, \boldsymbol{\delta}\mathbf{u}_{k+i} \\ & \boldsymbol{\delta}\mathbf{u}_{k+i} \in \mathbb{U}_\delta \\ & \mathbf{C}_v\, \boldsymbol{\zeta}_{k+i} \in \mathbb{Q} \times \mathbb{U} \\ & \boldsymbol{\zeta}_{k+N} \in \mathbb{O}_\epsilon. \end{array} \tag{7}$$

The first constraint defines the velocity form system dynamics. The second constraint imposes limitations on the maximum and minimum *increment* of the input $\mathbf{u}$. The third constraint is equivalent to requiring that $\mathbf{q}_{k+i} \in \mathbb{Q}$, $\forall i = 0, \ldots, N-1$ and $\mathbf{u}_{k+i} \in \mathbb{U}$, $\forall i = 1, \ldots, N-2$, being matrix $\mathbf{C}_v \in \mathbb{R}^{2n\times 3n}$ such that (see ref. [23])

$$\begin{bmatrix} \mathbf{q}_k \\ \mathbf{u}_{k-1} \end{bmatrix} = \mathbf{C}_v\, \boldsymbol{\zeta}_k = \begin{bmatrix} \mathbf{A} & \mathbf{B} \\ \mathbf{0} & \mathbf{I}_n \end{bmatrix} \mathbf{S}_v^{-1} \begin{bmatrix} \mathbf{I}_n & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{I}_n & \mathbf{I}_n \end{bmatrix} \begin{bmatrix} \boldsymbol{\xi}_k \\ \bar{\mathbf{q}}_k \end{bmatrix},$$

with

$$\mathbf{S}_v = \begin{bmatrix} \mathbf{A} - \mathbf{I}_n & \mathbf{B} \\ \mathbf{A} & \mathbf{B} \end{bmatrix}, \quad \mathbf{C}_v = [\mathbf{C}_\xi,\ \mathbf{C}_q],$$

where $\mathbf{C}_\xi \in \mathbb{R}^{2n\times 2n}$ and $\mathbf{C}_q \in \mathbb{R}^{2n\times n}$. Finally, the last constraint imposes the terminal constraint on the state, and the feasibility constraint on $\bar{\mathbf{q}}_k$.

The outputs of the optimization (7) are both the control input increment $\boldsymbol{\delta}\mathbf{U}_k$ that steers the state toward the artificial set-point $\bar{\mathbf{q}}_k$, and the value of $\bar{\mathbf{q}}_k$ itself. In particular, provided that $\mathbf{P}$, $\mathbf{T}$ and $\mathbb{O}_\epsilon$ are chosen as explained in ref. [23], it can be demonstrated that if at time $k = 0$ a solution to (7) exists, then the resulting RH MPC law asymptotically steers $\mathbf{q}_k$ to the *real* set point $\mathbf{q}_k^d$ when feasible, or to its closest feasible point $\bar{\mathbf{q}}_k$ when $\mathbf{q}_k^d$ is unfeasible, with

$$\bar{\mathbf{q}}_k = \arg \begin{array}{cl} \min\limits_{\mathbf{q}} & \|\mathbf{q} - \bar{\mathbf{q}}_k\|^2_{\mathbf{T}} \\ \text{s.t.} & \mathbf{C}_q\, \mathbf{q} \in \mathbb{Q}_\epsilon \times \mathbb{U}_\epsilon \end{array}. \tag{8}$$

$\mathbb{Q}_\epsilon$ and $\mathbb{U}_\epsilon$ are subsets of $\mathbb{Q}$ and $\mathbb{U}$, that is, with $\epsilon \geq 0$:

$$\begin{aligned} \mathbb{Q}_\epsilon &= \{\mathbf{q} \in \mathbb{R}^n : \ \mathbf{G}_q\, \mathbf{q} \leq \mathbf{b}_q - \epsilon\} \subseteq \mathbb{Q} \\ \mathbb{U}_\epsilon &= \{\mathbf{u} \in \mathbb{R}^n : \ \mathbf{G}_u\, \mathbf{u} \leq \mathbf{b}_u - \epsilon\} \subseteq \mathbb{U} \end{aligned} \tag{9}$$

Constraints on $\mathbf{q}$ and $\mathbf{u}$ are also fulfilled, and offset-free tracking is achieved thanks to the velocity form formulation.

3. On-Line Safe Vortex Fields for Improved Navigation

To improve autonomous navigation in unknown environments, besides the reference $\mathbf{q}_k^d$, it can be useful to add an additional "evasive" reference $\mathbf{q}_k^e$ for the robot state $\mathbf{q}_k$. In particular, $\mathbf{q}_k^e$ should be defined to help the robot mobile base circumnavigate possible obstacles without getting stuck in local minima.

3.1. Potential field rotation algorithm

The classical artificial vortex field formulation[28] can be exploited. Briefly, a vortex field F^v is an artificial potential field depending on obstacles configuration, and its gradient ∇F^v is a vector that produces a rotation around the considered obstacles. In particular, as artificial field we use the Danger Field,[29] elaborating a procedure to compute such field based on distance sensors information and to suitably rotate it, so as to obtain an actual vortex field.

The Danger Field assigns a danger level and the maximum danger growth direction to each point of the space surrounding a possibly moving robot, considering the robot as the source of danger. In particular, the Danger Field evaluated in a given point decreases with the distance of such point from the robot, whereas it increases with the amplitude of the relative velocity, more so if the robot is moving fast toward the considered position. The vector field $\overrightarrow{DF}$ can be easily defined in a position $\mathbf{r}$ as

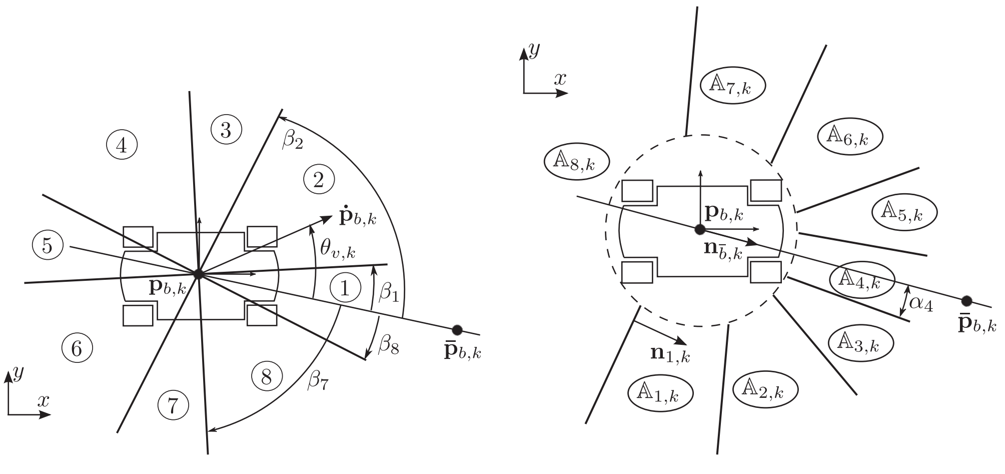
$$\overrightarrow{DF}(\mathbf{r}, \mathbf{q}, \dot{\mathbf{q}}) = DF(\mathbf{r}, \mathbf{q}, \dot{\mathbf{q}}) \frac{\nabla DF(\mathbf{r}, \mathbf{q}, \dot{\mathbf{q}})}{\|\nabla DF(\mathbf{r}, \mathbf{q}, \dot{\mathbf{q}})\|}, \tag{10}$$

where DF is the Danger Field value and ∇DF its gradient. Thus, $\overrightarrow{DF}(\mathbf{r}, \mathbf{q}, \dot{\mathbf{q}})$ is a vector anchored in $\mathbf{r}$, with the intensity $DF(\mathbf{r}, \mathbf{q}, \dot{\mathbf{q}})$, pointing in the $\nabla DF(\mathbf{r}, \mathbf{q}, \dot{\mathbf{q}})$ direction.

Consider now the mobile manipulator platform endowed with a number of exteroceptive sensors, able to perceive the environment surrounding the robot and to return a measurement that can be converted into the 3D positions of the detected objects (e.g., distance sensors or depth sensors). Let us define $\mathcal{P}_{o,k}$ as the set containing all the detected obstacle positions $\mathbf{r}_{oj}$ at time instant k, with $\dim(\mathcal{P}_{o,k}) = n_o$. Defining $DF_j = DF(\mathbf{r}_{oj}, \mathbf{q}, \dot{\mathbf{q}})$ and $\overrightarrow{DF}_j = \overrightarrow{DF}(\mathbf{r}_{oj}, \mathbf{q}, \dot{\mathbf{q}})$, the overall repulsive Danger Field vector produced by the platform can be defined as

$$\overrightarrow{DF}_r(\mathbf{q}, \dot{\mathbf{q}}) = \max_j\{DF_j\} \, (\sum_j \overrightarrow{DF}_j)/\| \sum_j \overrightarrow{DF}_j \|. \tag{11}$$

$\overrightarrow{DF}_r(\mathbf{q}, \dot{\mathbf{q}})$ is a vector whose direction is the one of maximum danger increment when considering all the n_o points, and having the modulus of the Danger Field computed in the point that is in the most



(a) Motion State (b) Obstacle occupancy

Fig. 1. (a) Motion State and (b) Occupancy Map.

dangerous configuration. Note that, knowing the platform position and velocity and the positions of $\mathbf{r}_{oj}$, the evaluation of $\overrightarrow{DF}_r$ in closed form is straightforward (see ref. [29]).

Moving the robot in the direction defined by $\overrightarrow{DF}_r(\mathbf{q}, \dot{\mathbf{q}})$ would clearly produce the maximal danger reduction for the considered obstacles configuration. However, as it is common when using repulsive artificial fields for mobile robots, this evasive motion could be in contrast with the desired motion toward the goal configuration $\mathbf{q}^d$, resulting in a local minimum standstill situation. The vector $\overrightarrow{DF}_r(\mathbf{q}, \dot{\mathbf{q}})$ must then be properly rotated, obtaining an actual vortex field, that helps mitigating this issue. Note in particular that a map of the environment is not available; therefore, the best rotation strategy must be evaluated at runtime based only on sensors perceptions. An algorithm has then been developed that rotates each component $\overrightarrow{DF}_j$ of 0 or $\pm 90^\circ$, depending on the current platform position $\mathbf{r}_{b,k} = [x_{b,k}, y_{b,k}, z_b]^{\mathrm{T}}$ (z_b is the constant vertical position of the platform center of gravity), on its velocity $\dot{\mathbf{r}}_{b,k} = [\dot{x}_{b,k}, \dot{y}_{b,k}, 0]^{\mathrm{T}}$, on the current goal configuration $\mathbf{r}^d_{b,k} = [x^d_{b,k}, y^d_{b,k}, z_b]^{\mathrm{T}}$ and on the remaining perceived obstacles positions $\mathbf{r}_{oi} \in \mathcal{P}_{o,k}$, $i \neq j$. Once the rotated vectors $\overrightarrow{DF}_{vj}$ are computed, the overall Vortex Danger Field vector is defined as

$$\overrightarrow{DF}_v = \max_j \{DF_j\} \left(\sum_j \overrightarrow{DF}_{vj}\right) / \left(\| \sum_j \overrightarrow{DF}_{vj} \|\right). \tag{12}$$

The main rationale behind the rotation algorithm is to rotate each $\overrightarrow{DF}(\mathbf{r}_{oj}, \mathbf{q}, \dot{\mathbf{q}})$ so that the angle between the planar projection of $\overrightarrow{DF}_{vj}$ and the planar projection of the segment $(\mathbf{r}^d_{b,k} - \mathbf{r}_{b,k})$ is the smallest possible one, whenever this condition is not in contrast with the remaining obstacles configuration at current time. In this way, the platform movement will be deviated as little as possible from the straight line $(\mathbf{r}^d_{b,k} - \mathbf{r}_{b,k})$. Because no map of the environment is available, it is necessary to also account for the environment configuration at the previous time instant(s). This is done to avoid "forgetting" obstacles that are no longer detected by the sensors (e.g., because they are out of the sensors measurement range), but may actually have not moved. The information about the previous obstacle configuration is derived from the current platform velocity $\dot{\mathbf{p}}_{b,k}$, that is the result of the algorithm at the previous time instant, and it is then used as a memory of the previous environment state.

The platform state of motion is subdivided into eight possible Motion States, depending on the angle $\theta_{v,k}$ between the platform velocity and the platform main axis (whose orientation is θ_{bk}), as shown in Fig. 1.

Moreover, the space surrounding the robot is subdivided into $n_A = 8$ regions, defined by n_A planes, that are attached to the platform and therefore depend on $\mathbf{q}_{b,k}$ (the 2D projection of such planes is

Algorithm 1: Gradient Rotation (see Fig. 1)

```
Input: $\mathbf{q}_{b,k}, \dot{\mathbf{q}}_{b,k}, \mathbf{r}_{b,k}, \dot{\mathbf{r}}_{b,k}, \mathbf{P}_{ok}, \mathbf{r}^d_{b,k}, d_u$ (sensor range), $d_l$ (threshold, $d_l < d_u$)
Result: Rotated Danger Field $\overrightarrow{DF}_v$
begin
    compute $MotionState$
    compute $\mathbb{A}_{ik}, i = 1, \dots, n_A$
    compute $O(\mathbb{A}_{ik}, \mathcal{P}_{o,k}, d_u), i = 1, \dots, n_A$
    compute $O(\mathbb{A}_{ik}, \mathcal{P}_{o,k}, d_l), i = 1, \dots, n_A$
    if (Motion State=1) then
        ...
        ...
    else if (Motion State=3) then
        foreach $\mathbf{r}_{oj} \in \mathcal{P}_{o,k}$ do
            $\overrightarrow{DF}_j \leftarrow \overrightarrow{DF}(\mathbf{r}_{oj}, \mathbf{r}_{b,k}, \dot{\mathbf{r}}_{b,k})$
            if $\mathbf{r}_{oj} \in \mathbb{A}_2 \cup \mathbb{A}_3 \cup \mathbb{A}_4$ then Fig. 2, case 1
                rotate $\overrightarrow{DF}_j$ of $-90^\circ$
            else if $\mathbf{r}_{oj} \in \mathbb{A}_5$ then
                if $O(\mathbb{A}_5, \mathcal{P}_{o,k}, d_l) \wedge O(\mathbb{A}_3, \mathcal{P}_{o,k}, d_u)$ then Fig. 2, case 1
                    rotate $\overrightarrow{DF}_j$ of $-90^\circ$
                else Fig. 2, case 2
                    rotate $\overrightarrow{DF}_j$ of $90^\circ$
                end
            else
                set $\overrightarrow{DF}_j \leftarrow [0, 0, 0]^{\mathrm{T}}$
            end
        end
        compute $\overrightarrow{DF}_v$ as in Eq. (12)
    else
        ...
    end
```

reported in Fig. 1). With reference to Fig. 1, let us define for $i = 1, \dots, n_A$ the following unitary vectors:

$$\mathbf{n}_{i,k} = [\cos(\theta_{b,k} - \alpha_i + \pi/2), \sin(\theta_{b,k} - \alpha_i + \pi/2), 0]^{\mathrm{T}}.$$

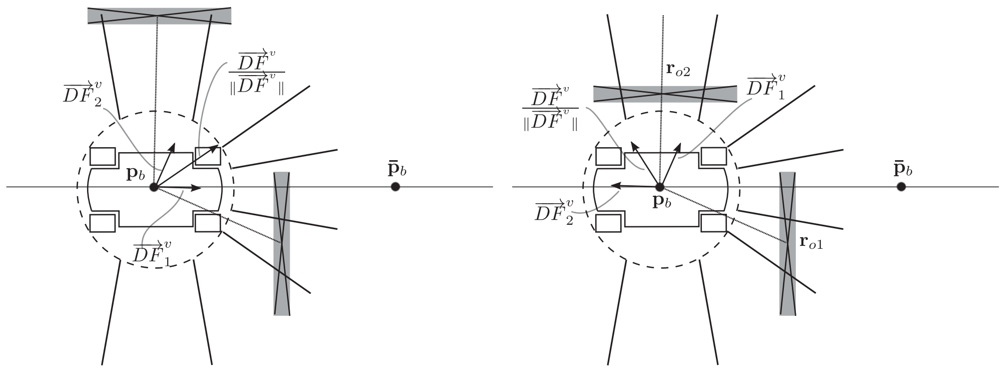
Each region $\mathbb{A}_{i,k},\ i = 1, \dots, n_A$ is then defined as

$$\mathbb{A}_{i,k} = \{\mathbf{r} \in \mathbb{R}^3 : \begin{array}{l} \mathbf{n}^{\mathrm{T}}_{i,k} \cdot (\mathbf{r} - \mathbf{r}_{b,k}) \geq 0, \\ \mathbf{n}^{\mathrm{T}}_{i+1,k} \cdot (\mathbf{r} - \mathbf{r}_{b,k}) < 0\}. \end{array} \tag{13}$$

Given the detected obstacles point $\mathbf{r}_{oj}$ and a threshold distance values d (that could be the sensor range or a specified dangerous threshold), the $Occupancy$ function is defined as

$$O(\mathbb{A}_{ik}, \mathcal{P}_{o,k}, d) = \begin{cases} 1 & \text{if } \exists\, \mathbf{r}_{oj} \in \mathbb{A}_{i,k} \cap \mathcal{P}_{o,k} : \|\mathbf{r}_{oj} - \mathbf{r}_{bk}\| \leq \bar{d} \\ 0 & \text{otherwise,} \end{cases} \tag{14}$$

and it helps define an Occupancy Map. Given the Motion State and the Occupancy Map, the rotation strategy can be defined. Part of the rotation algorithm is reported in Algorithm 1 and exemplified in Fig. 2.


$\overrightarrow{DF}^v_2$
$\frac{\overrightarrow{DF}^v}{\|\overrightarrow{DF}^v\|}$
$\mathbf{p}_b$
$\bar{\mathbf{p}}_b$
$\overrightarrow{DF}^v_1$
$\mathbf{r}_{o2}$
$\mathbf{r}_{o1}$

(a) First possible situation (b) Second possible situation

Fig. 2. Algorithm effect for Motion State = 2. (a) First possible situation. (b) Second possible situation.

3.2. Inclusion of the evasive motion in the MPC cost function

The planar projection of $\overrightarrow{DF}_v = [x_v, y_v, z_v]^{\mathrm{T}}$ computed at time instant k can be interpreted as a desired (rotational) evasive motion for the robot platform, that would allow the robot to circumnavigate all the obstacles. In the proposed MPC approach, $\overrightarrow{DF}_v$ is included in the cost function L by modifying the reference $\mathbf{q}^d_k$ as follows. Define the evasive reference vector $\mathbf{q}^e_k = [x_{b,k} + x_v, y_{b,k} + y_v, \theta_{b,k}, \mathbf{q}^{\mathrm{T}}_{a,k}]^{\mathrm{T}} \in \mathbb{R}^n$. A new reference is then defined as

$$\mathbf{q}^{de}_k = m\mathbf{q}^d_k + (1 - m)\mathbf{q}^e_k, \tag{15}$$

where $m = m(\|\overrightarrow{DF}_{v,k}\|) \in \mathbb{R}$ is a scalar value depending on the field gradient modulus as

$$m = \begin{cases} 1 & \text{if } \|\overrightarrow{DF}_v\| \leq DF_{th} \\ 0 & \text{otherwise.} \end{cases}$$

The new reference $\mathbf{q}^{de}_k$ substitutes $\mathbf{q}^d_k$ in Eq. (6). m assures that local minima are avoided, because at a given instant k only one of the two possibly conflicting goals $\mathbf{q}^d$ and $\mathbf{q}^e$ is considered in L. When the field modulus is below the threshold DF_{th} (e.g., when obstacles are far from the robot, or the platform is moving slowly), only $\mathbf{q}^d$ is considered. When instead $\|\overrightarrow{DF}_v\| > DF_{th}$, only the evasive reference $\mathbf{q}^e$ is considered in L. Note that such a switch of the problem goal is completely compatible with the MPC formulation presented in Section 2.

Finally, note that $\overrightarrow{DF}_v$ could be used in different ways, e.g., by directly including it as a term to be penalized in L, making the cost function non-quadratic and non-convex. The proposed solution, which includes its gradient as an evasive motion reference, instead, maintains the linear quadratic formulation of the problem, thus retaining the necessary computational performance.

Summarizing, by replacing $\mathbf{q}^d_k$ with $\mathbf{q}^{de}_k$ in the cost function of (6), the solution of Eq. (7) drives the robot around obstacles when necessary, while moving toward $\mathbf{q}^d_k$ otherwise.

4. Cartesian Space Formulation

As of now, the control problem has been defined in the joint space. It is common, however, to define a robot task in term of the desired end-effector Cartesian position and orientation. In this section, a methodology to include Cartesian references in the MPC problem is presented.

Let us define an end-effector pose $\mathbf{P} = [\mathbf{r}^{\mathrm{T}}_{ee}, \boldsymbol{\epsilon}^{\mathrm{T}}_{ee}]^T \in \mathbb{R}^6$ as a vector composed of the Cartesian end-effector position $\mathbf{r}_{ee} = [x, y, z]^{\mathrm{T}}_{ee} \in \mathbb{R}^3$ and of a minimal representation of its orientation $\boldsymbol{\epsilon}_{ee} = [\phi, \chi, \psi]^{\mathrm{T}}_{ee} \in \mathbb{R}^3$ (in particular, the rotations around the x, y, z axis of the global reference frame). The non-linear forward kinematics function $fk(\cdot) : \mathbb{R}^n \to \mathbb{R}^6$ computes the pose $\mathbf{P}$ corresponding to $\mathbf{q}$ as $\mathbf{P} = fk(\mathbf{q})$.

The MPC problem with Cartesian space reference is then defined as finding the control inputs sequence $\boldsymbol{\delta}\mathbf{U}_k$ that moves the end-effector from its current pose $\mathbf{P}$ to a possibly time varying (piecewise constant) pose $\mathbf{P}_k^d = [(\mathbf{r}_k^d)^{\mathrm{T}}, (\boldsymbol{\epsilon}_k^d)^{\mathrm{T}}]^{\mathrm{T}}$, if it is feasible in terms of joint variables, while complying with the imposed constraints of Section 2.

One possibility is to solve the inverse kinematics problem for the considered pose $\mathbf{P}_k^d$, finding the corresponding desired state vector $\mathbf{q}_k^d$, and thus returning to the problem of Section 2. For non-redundant robots, the inverse kinematics problem can be usually solved in closed form, through an inverse kinematics function $ik(\cdot) = fk^{-1}(\cdot) : \mathbb{R}^6 \to \mathbb{R}^6$, so that $\mathbf{q}_k^d = ik(\mathbf{P}_k^d)$. However, mobile manipulators typically have more than 6 DoF, namely possessing $n_\rho = n - 6$ degrees of redundancy. In this case, ik provides infinite solutions for the same $\mathbf{P}$, i.e., the same configuration $\mathbf{P}$ corresponds to an infinity of state vectors $\mathbf{q}$.

Two possibility then exist for the intended approach. On the one hand, it is possible to augment the task space (see refs. [30, 31]) by including n_ρ tasks to be fulfilled by the robot besides the end-effector position and orientation task. This can be done, e.g., by defining a vector of redundancy parameters $\boldsymbol{\rho}(\mathbf{q}) = [\rho_1(\mathbf{q}), \ldots, \rho_{n_\rho}(\mathbf{q})]^{\mathrm{T}}$ that depends on the joint configuration $\mathbf{q}$. Suitable desired values $\boldsymbol{\rho}^d \in \mathbb{R}^{n_\rho}$ can then be chosen, and $\boldsymbol{\rho}(\mathbf{q}) = \boldsymbol{\rho}^d$ can be used as the additional task. An inverse kinematics function $ik_\rho(\cdot) : \mathbb{R}^{6+n_\rho} \to \mathbb{R}^n$ can thus be derived, so that $\mathbf{q}^d = ik_\rho([(\mathbf{P}^d)^{\mathrm{T}}, (\boldsymbol{\rho}^d)^{\mathrm{T}}]^{\mathrm{T}})$ exists in closed form. Redundancy parameters are particularly useful, because they give a direct physical interpretation of the robot redundancy (see the appendix, where a set of redundancy parameters $\boldsymbol{\rho} = [\rho_1, \rho_2, \rho_3]$ (Fig. 13) is introduced for solving the inverse kinematics of a mobile manipulator endowed with a 6 DoF anthropomorphic arm with spherical wrist). We exploited this particular approach in the experiments presented in Section 6.

On the other hand, however, it is not always easy to identify suited redundancy parameters for a given robot. In this case, a Closed Loop Inverse Kinematics (CLIK) algorithm (see, e.g., ref. [32]) can for example be used to solve the inverse kinematics problem, computing at each control cycle a reference $\mathbf{q}^d$ that iteratively converges to $\mathbf{P}^d = fk(\mathbf{q}^d)$.

Finally, note that a third different approach to the Cartesian problem would be to directly replace the set-point error of Eq. (4) with $\mathbf{e}_k = \bar{\mathbf{P}}_k - \mathbf{P}_k^d$, and the term $\bar{\mathbf{q}}_k - \mathbf{q}_k^d$ in the cost function of Eq. (6) with $\bar{\mathbf{P}}_k - \mathbf{P}_k^d$, $\bar{\mathbf{P}}_k$ being a new optimization variable instead of $\bar{\mathbf{q}}_k$. In this case, however, the direct kinematics $fk(\cdot)$ must be added as a constraint: L would no longer be quadratic with respect to the decision variables, and the problem would lose its QP formulation, seriously increasing the computational load for the numerical solver, especially for longer prediction horizons. It should be noted in any case that recent ad-hoc implementations (see, e.g., ref. [8]) allow for the on-line resolution of non-linear MPC. However, as explained in Section 2, in this paper we concentrated on high-level trajectory planning, thus considering a purely kinematic model of the robot. With this in mind, the added burden of developing a tailored solution algorithm for the resolution of the non-linear problem does not appear to offer significant advantages with respect to the more agile and compact implementation of a QP problem, which has then been considered in the remaining of this work.

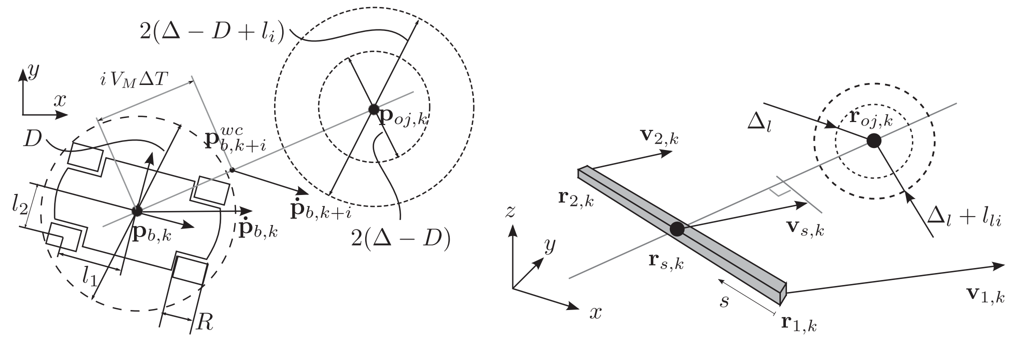
5. Constraints Formulation

Once the MPC problem for the joint space and for the Cartesian space has been set, the problem constraints must be defined, specifying $\mathbb{Q}$, $\mathbb{U}$ and $\mathbb{U}_\delta$ of Eqs. (2) and (7). We will define polytopic constraints sets, i.e., expressed through a matrix inequality of the form $\mathbf{Gu} \leq \mathbf{b}$, that is linear with respect to $\mathbf{u}$. The resulting MPC problem (7) is then a linear-quadratic one, that can be efficiently solved online with QP numerical methods. In particular, the following constraints are enforced: velocity bounds for the mobile base and arm, position limits for the arm joints, acceleration bounds for each joint, collision avoidance constraints for both the base and the arm, and FoV constraints.

5.1. Linear constraints over the prediction horizon

First note that the evolution of the system on the prediction horizon can be expressed as

$$\mathbf{q}_{k+i} = \mathbf{A}^i \mathbf{q}_k + \sum_{j=0}^{i-1} \mathbf{A}^{i-j-1} \mathbf{B}\, \mathbf{u}_{k+j}, \tag{16}$$



(a) Robot mobile base. (b) Generic robot link

Fig. 3. Collision avoidance constraint for a mobile manipulator. (a) Robot mobile base. (b) Generic robot link.

so that the constraints on the state variables $\mathbf{q}$ can be expressed as linear constraints on the input variable sequence $\mathbf{U}_k = [\mathbf{u}_k^{\mathrm{T}}, \mathbf{u}_{k+1}^{\mathrm{T}}, \ldots, \mathbf{u}_{k+N-1}^{\mathrm{T}}]^{\mathrm{T}}$ on the whole prediction horizon. $\mathbf{u}_{k+i}$ can be computed from the sequence $\boldsymbol{\delta}\mathbf{U}_k$ as

$$\mathbf{u}_{k+i} = \mathbf{u}_{k-1} + \sum_{j=0}^{i} \boldsymbol{\delta}\mathbf{u}_{k+j}. \tag{17}$$

The first set of constraints limit the translational and rotational velocity for the robot base and the rotational velocity for the arm joints. They can be simply expressed as

$$|\mathbf{E}_a\, \mathbf{u}_{k+i}| \leq \dot{\mathbf{q}}_a^{\max}. \tag{18}$$

where $\mathbf{E}_a \in \mathbb{R}^{n_a \times n}$ is such that $\mathbf{q}_a = \mathbf{E}_a\mathbf{q}$.

The velocity limits $\dot{\mathbf{q}}_b^{\max}$ for the mobile base may not be immediately available, but they should instead be derived from the speed limits of the platform wheels. For the common case of omnidirectional platforms endowed with Mecanum wheels, with reference to Fig. 3(a), the platform kinematics yields

$$\boldsymbol{\omega}_w = \frac{1}{R}\begin{bmatrix} 1 & 1 & -(l_1+l_2) \\ 1 & -1 & (l_1+l_2) \\ 1 & -1 & -(l_1+l_2) \\ 1 & 1 & (l_1+l_2) \end{bmatrix} \dot{\mathbf{q}}_b = \mathbf{L}\dot{\mathbf{q}}_b = \mathbf{L}\mathbf{E}_b\dot{\mathbf{q}}.$$

where R, l_1 and l_2 are reported in Fig. 3(a), $\boldsymbol{\omega}_w \in \mathbb{R}^4$ is a vector containing the four Mecanum wheels spinning velocities and $\mathbf{E}_b \in \mathbb{R}^{3\times n}$ is an extraction matrix that satisfies $\mathbf{q}_b = \mathbf{E}_b\mathbf{q}$. Thus, to constrain the platform velocities, the following inequalities can be defined:

$$-\boldsymbol{\omega}_w^{\max} \leq \mathbf{L}\mathbf{E}_b\mathbf{u}_{k+i} \leq \boldsymbol{\omega}_w^{\max}. \tag{19}$$

The second set of constraints, concerning the arm joints position limits, is written as

$$\mathbf{q}_a^{\min} \leq \mathbf{E}_a(\mathbf{A}\mathbf{q}_{k+i} + \mathbf{B}\mathbf{u}_{k+i}) \leq \mathbf{q}_a^{\max}. \tag{20}$$

Acceleration constraints can be enforced by approximating $\ddot{\mathbf{q}}_k = \dot{\mathbf{u}}_k \approxeq (\mathbf{u}_k - \mathbf{u}_{k-1})/\Delta T = \boldsymbol{\delta}\mathbf{u}/\Delta T$, so that

$$|\boldsymbol{\delta}\mathbf{u}_{k+i}| \leq \Delta T \ddot{\mathbf{q}}^{\max}, \tag{21}$$

is the linear constraint to be imposed on the decision variable $\boldsymbol{\delta}\mathbf{u}$, and defines the set $\mathbb{U}_\delta$ in Eq. (7). Note that constraints (18)–(21) are easily enforced on the whole prediction horizon, being always linear with respect to $\mathbf{u}_{k+i}$ or $\boldsymbol{\delta}\mathbf{u}_{k+i}$, $\forall i$.

5.2. Non-linear constraints over the prediction horizon

Differently from the previous case, there might exist constraints whose matrices $\mathbf{G}_{k+i} = \mathbf{G}(\mathbf{q}_{k+i})$ non-linearly depend on $\mathbf{q}$, i.e., $\mathbf{G}$ is a generic non-linear function of $\mathbf{q}$. This means that these constraints are non-linear with respect to $\mathbf{u}$ when $i > 0$. In fact, at time instant $k+i$, $\mathbf{G}$ depends non-linearly on $\mathbf{q}_{k+i}$, that is linear in $\mathbf{u}_k, \ldots, \mathbf{u}_{k+i-1}$. Hence, $\mathbf{G}_{k+i}$ depends non-linearly on $\mathbf{u}_k, \ldots, \mathbf{u}_{k+i-1}$, and the constraint $\mathbf{G}_{k+i}\mathbf{u}_{k+i}$ is non-linear in the control input sequence.

A possible way to maintain linearity is to exploit the MPC recursive nature. At time k, in fact, $\mathbf{q}_{k+i}$, $i = 1, \ldots, N$ can be approximated as $\tilde{\mathbf{q}}_{k+i}$, that is computed through Eq. (16) using as input the sequence $\mathbf{U}_{k-1} = [\mathbf{u}_{k-1}^{\mathrm{T}}, \mathbf{u}_k^{\mathrm{T}}, \ldots, \mathbf{u}_{k+N-2}^{\mathrm{T}}]^{\mathrm{T}}$. $\mathbf{U}_{k-1}$ is computed through Eq. (17) from the solution $\boldsymbol{\delta}\mathbf{U}_{k-1}$ of the previous controller iteration. It is then possible to approximate $\mathbf{G}(\mathbf{q}_{k+i}) \approxeq \mathbf{G}(\tilde{\mathbf{q}}_{k+i})$. In this way, the constraint $\mathbf{G}(\tilde{\mathbf{q}}_{k+i})\dot{\mathbf{q}} \leq \mathbf{b}$ is again linear in the control input sequence.

This approximation is evidently less accurate whenever major modifications occur either in the environment (e.g., new obstacles appear) or in the task (a new value of $\mathbf{P}^d$ is set). In these situations, a "worst case" approach has been defined to compute suitable values of $\mathbf{G}(\mathbf{q}_{k+i})$, as will be detailed in the following for the different constraints.

5.2.1. Base collision avoidance. This constraint is used to avoid collisions of the mobile base with perceived and possibly known obstacles (note that the rotational field of Section 3 does not guarantee collision avoidance, not being repulsive). The approach presented in ref. [19] and inherited from ref. [33] is here developed, extending such constraints on the whole MPC prediction horizon.

With reference to Fig. 3(a), consider one of the n_o perceived obstacles, located in $\mathbf{r}_{oj} \in \mathcal{P}_o$, and the mobile base at position $\mathbf{p}_b = [x, y]_b^{\mathrm{T}}$ having velocity $\dot{\mathbf{p}}_b$. Let $\mathbf{p}_{oj} = [r_{ojx}, r_{ojy}]^{\mathrm{T}}$ be the projection of $\mathbf{r}_{oj}$ on the x–y plane, and $\mathbf{E}'_b \in \mathbb{R}^{2\times n}$ be an extraction matrix, so that $\mathbf{p}_b = \mathbf{E}'_b\mathbf{q}$. Based on ISO/ANSI safety standards, in order to avoid collisions, the robot motion is restricted through the inequality $V \cdot T_s \leq \max(0,\ d - \Delta)$, where V is the signed component of the robot velocity toward the obstacle, T_s is the worst case robot stopping time, d is the robot–obstacle distance and $\Delta > 0$ is a clearance parameter. As explained in ref. [19], for $j = 1, \ldots, n_o$, at time instant k such obstacle avoidance constraint can be arranged as

$$T_s \mathbf{d}_{j,k}^T \mathbf{E}'_b \mathbf{u}_k \leq \left(\max(0, \|\mathbf{d}_{j,k}\| - \Delta)\right)^2, \tag{22}$$

where $\mathbf{d}_{j,k} = \mathbf{p}_{oj,k} - \mathbf{p}_{b,k}$. $\max(0, \|\mathbf{d}_{j,k}\| - \Delta)$ denotes the distance between a circle having radius D built around the robot base, and a circle centred in $\mathbf{p}_{oj}$ with radius $(\Delta - D)$.

This constraint does not enforce a minimum distance between the robot and the obstacle: It is in fact possible that $d_m = \min_{j=1,\ldots,n_o} \|\mathbf{d}_{j,k}\| < \Delta$, e.g., if an obstacle that is closer to the robot than Δ suddenly appears. In this case, the constraint will forbid any further robot motion toward the obstacle, ensuring collision avoidance. Note that n_o can be very high, e.g., if depth sensors are used to perceive the environment. In this case, a clustering technique should be used to group the detected points into a smaller number of obstacles. The collision avoidance constraint can then be enforced for the closest point of each cluster to the robot, thus avoiding an explosion of the problem constraints. Alternatively, in ref. [34], it is shown how to formalize this constraint for the convex hull containing the clustered obstacle points.

The "worst case" situation for the considered constraint can now be derived, so as to extend this constraint on the prediction horizon whenever $\mathbf{q}_{k+i}$ cannot be properly approximated by $\tilde{\mathbf{q}}_{k+i}$. The worst case situation occurs when the robot base and the obstacle move at maximum speed V_M one toward the other (see Fig. 3(a)), so that at time instant $k+i$ the robot is in position $\mathbf{p}_{b,k+i}^{wc}$. Following a procedure that is similar to the one reported in ref. [19], after some manipulation the worst case obstacle avoidance constraint at time instant $k+i$, $i = 0, \ldots, N-1$ can be expressed as

$$T_s \mathbf{d}_{j,k}^T \mathbf{E}'_b \mathbf{u}_{k+i} \leq (\max(0, \|\mathbf{d}_{j,k}\| - (l_i + \Delta)))^2, \tag{23}$$

where $l_i = i\, V_M \Delta T$ expresses the space covered in i time steps by the robot base and the obstacle moving at maximum speed V_M toward each other.

Extending the obstacle avoidance constraint at time instant $k+i$ is therefore equivalent to increasing the clearing distance Δ by the quantity l_i, that accounts for the worst case relative motion

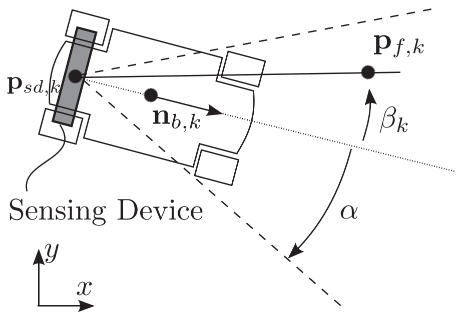


Fig. 4. Field-of-view constraint for a generic sensing device.

between the robot and the obstacle. When considering n_o obstacles, constraint (23) can be rewritten as $\mathbf{G}_b(\mathbf{q}_k)\mathbf{u}_{k+i} \leq \mathbf{b}_{b,k+i}$, with $\mathbf{G}_b \in \mathbb{R}^{n_o \times n}$ and $\mathbf{b}_b \in \mathbb{R}^{n_o}$, that is linear with respect to $\mathbf{u}(k+i)$.

5.2.2. Arm collision avoidance. Collision avoidance constraints can be applied also to the robotic arm of the mobile manipulator, so as to avoid impacts of the arm with perceived or possibly known obstacles in the environment, and with the mobile base itself. Again with reference to ref. [33] and to Fig. 3(b), the collision avoidance constraints of Section 5.2.1 can be formulated in the 3D Cartesian space for a generic robot link as the pair

$$T_s \mathbf{d}_{1o,k}^{\mathrm{T}} \mathbf{v}_{1,k} \leq (\max(0, d_{lj,k} - \Delta_l))^2$$

$$T_s(\mathbf{d}_{1o,k}^{\mathrm{T}} \mathbf{v}_{2,k} - \mathbf{d}_{12,k}^{\mathrm{T}} \mathbf{v}_{1,k}) \leq (\max(0, d_{lj,k} - \Delta_l))^2,$$

where

$$\begin{cases} \mathbf{d}_{1o,k} = \mathbf{r}_{oj,k} - \mathbf{r}_{1,k} \\ \mathbf{d}_{12,k} = \mathbf{r}_{2,k} - \mathbf{r}_{1,k} \\ d_{lj,k} = \min_s \|\mathbf{r}_{oj,k} - \mathbf{r}_{s,k}\|. \end{cases}$$

When it is positive, $d_{lj} - \Delta_l$ represents the distance between a sphere of radius Δ_l centred in $\mathbf{r}_{oj}$ and the segment from $\mathbf{r}_1$ to $\mathbf{r}_2$. Defining $\mathbf{J}_{1,k}$ and $\mathbf{J}_{2,k}$ as the position Jacobians of $\mathbf{r}_{1,k}$ and $\mathbf{r}_{2,k}$, and $b_{lj,k} = (\max(0, d_{lj,k} - \Delta_l))^2$, the above constraint can be formulated at time instant k as

$$\begin{bmatrix} T_s \mathbf{d}_{1o,k}^{\mathrm{T}} \mathbf{J}_{1,k} \\ T_s(\mathbf{d}_{1o,k}^{\mathrm{T}} \mathbf{J}_{2,k} - \mathbf{d}_{12,k}^{\mathrm{T}} \mathbf{J}_{1,k}) \end{bmatrix} \mathbf{u}_k \leq \begin{bmatrix} b_{lj,k} \\ b_{lj,k} \end{bmatrix} \tag{24}$$

that is linear with respect to $\mathbf{u}_k$. Note that the position Jacobians $\mathbf{J}_{1,k}$ and $\mathbf{J}_{2,k}$ take into account the effects of all the mobile manipulator (i.e., base and arm) velocities $\mathbf{u}_k$ to compute $\mathbf{v}_{1,k}$ and $\mathbf{v}_{2,k}$.

This constraint has been extended over the prediction horizon in the "worst case" scenario with a similar procedure to Section 5.2.1. Defining $l_{li} = i V_{lM} \Delta T$, $i = 0, \ldots, N-1$, the worst case situation has been tackled by modifying constraint (24) right-hand side as

$$b_{lj,k+i} = (\max(0, d_{lj,k} - (\Delta_l + l_{li}))^2. \tag{25}$$

Considering all n_o obstacles (with the same clustering consideration as Section 5.2.1), constraints (24) and (25) are rewritten as $\mathbf{G}_a(\mathbf{q}_k)\mathbf{u}_{k+i} \leq \mathbf{b}_{a,k+i}$, with $\mathbf{G}_a \in \mathbb{R}^{2n_o \times n}$ and $\mathbf{b}_a \in \mathbb{R}^{2n_o}$, that is linear with respect to $\mathbf{u}(k+i)$.

5.2.3. Field-of-view constraint. With reference to Fig. 4, suppose that the robot platform is endowed with a sensing device (e.g., a depth sensor or a camera) to track some features placed in a possibly time varying position $\mathbf{r}_{f,k}$. The FoV constraint is designed to keep $\mathbf{r}_{f,k}$ in the device FoV.

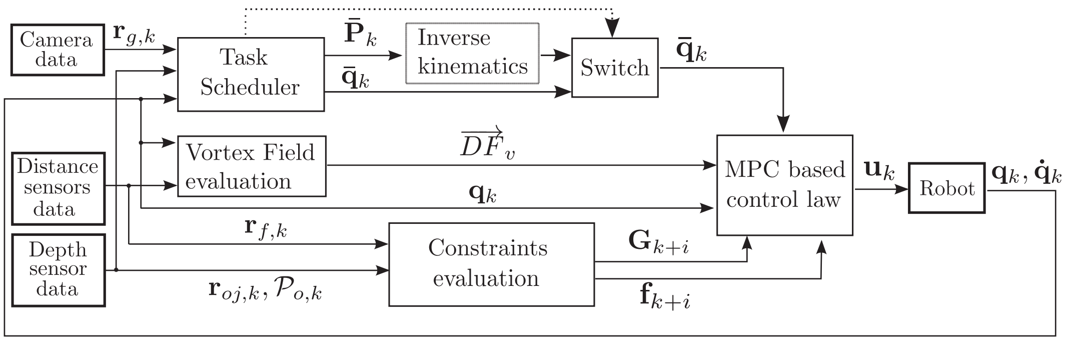


Fig. 5. Block diagram scheme of the controller.

Defining $\phi = 2\alpha < \pi$ as the FoV angle, and $\mathbf{p}_{f,k}$ as the x–y projection of $\mathbf{r}_{f,k}$, such constraint at step $k+1$ can be formulated as

$$|\beta_{k+1}| \le \alpha \Rightarrow \cos(\beta_{k+1}) \ge \cos(\alpha) = c_\alpha .$$

Since

$$\begin{cases} \cos(\beta_{k+1}) \approxeq \cos(\beta_k) + \Delta T\, \dot{\cos}(\beta_k) \\ \cos(\beta_k) = \dfrac{(\mathbf{p}_{f,k} - \mathbf{p}_{sd,k})^{\mathrm{T}} \mathbf{n}_{b,k}}{\|\mathbf{p}_{f,k} - \mathbf{p}_{sd,k}\|} = \dfrac{\mathbf{d}_{sd,k}^{\mathrm{T}}\, \mathbf{n}_{b,k}}{\|\mathbf{d}_{sd,k}\|}, \\ \dot{\cos}(\beta_k) = \left.\dfrac{\partial \cos(\beta)}{\partial \mathbf{q}}\right|_k \mathbf{u}_k = \mathbf{J}_{FoV,k}\, \mathbf{u}_k \end{cases}$$

the FoV constraint can be written as

$$1 \ge \frac{\mathbf{d}_{sd,k}^{\mathrm{T}}\, \mathbf{n}_{b,k}}{\|\mathbf{d}_{sd,k}\|} + \Delta T \mathbf{J}_{FoV,k}\, \mathbf{u}_k \ge c_\alpha . \tag{26}$$

This constraint is clearly linear with respect to $\mathbf{u}_k$, and thus it can be expressed in the form $\mathbf{G}_{FoV,k}\, \mathbf{u}_k \le \mathbf{b}_{FoV,k}$, where $\mathbf{G}_{FoV,k} \in \mathbb{R}^{1\times n}$ and $\mathbf{b}_{FoV}(k) \in \mathbb{R}$.

6. Experiments

The control scheme implementing the introduced MPC approach is reported in Fig. 5. A task scheduler manages the different phases of the robot task, that is divided into a sequence of subtasks defining the desired motion, either in the joint space as $\mathbf{q}_k^d$ or in the Cartesian space as $\mathbf{P}_k^d$. For the Cartesian space, $\mathbf{q}^d$ is computed through the inverse kinematics approach (Section 4). A new subtask is triggered by the scheduler whenever the error with respect to the desired set-point is small enough. During motion, all constraints described in Section 5 are enforced. The input of the controller are the robot state $\mathbf{q}_k$, the measured velocities (corresponding to $\mathbf{u}_{k-1}$) and the goal $\mathbf{q}_k^d$, as well as the perceived obstacles positions $\mathcal{P}_{o,k}$, the perceived feature position $\mathbf{r}_{f,k}$ and the perceived position $\mathbf{r}_{g,k}$ of objects to be grasped. The output of the controller consists of the velocity commands $\mathbf{u}_k = \boldsymbol{\delta}\mathbf{u}_k + \mathbf{u}_{k-1}$ (i.e., the first element of the MPC solution), which is sent to the lower level axis controllers. In the receding horizon fashion, the computation is iterated at every control cycle.

6.1. *Setup*

Our approach has been validated experimentally on a KUKA youBot mobile manipulator,[24] which possesses $n = 8$ DoF (3 for the omnidirectional platform and 5 for the arm) and can be velocity controlled. A PC runs the control code and manages communications with the different sensors and with the low-level axis controllers. The robot state and velocity are measured by proprioceptive sensors, including a built-in dead reckoning algorithm. Eleven inexpensive Sharp GP2Y0A02YK distance sensors fixed to the mobile base detect the obstacles. Their signals are acquired by an

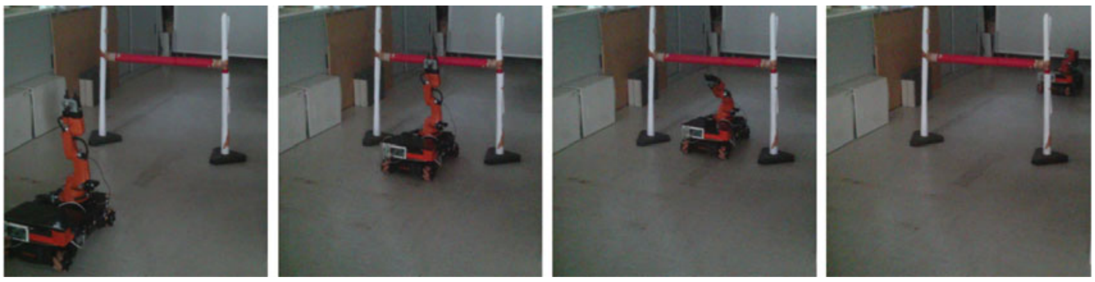


Fig. 6. Arm collision avoidance experiment snapshots.

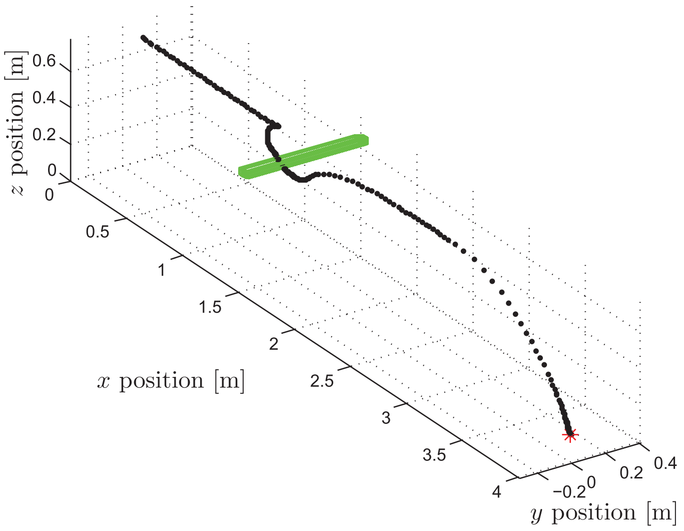


Fig. 7. Arm collision avoidance experiment. 3D plot of the end effector position (black dots), the goal position (red star) and the obstacle position (green lines) as tracked by the depth sensor.

Arduino board, interfaced with the PC. A camera is mounted on the end-effector, in order to detect and localize objects to be grasped. Images are acquired and processed by the PC exploiting OpenCV libraries.[35] An Asus Xtion depth sensor is also mounted on the platform to detect and track the desired feature (e.g., the hand of a human operator) and obstacles that cannot be detected by the distance sensors (e.g., if they do not touch the ground), thanks to the OpenNI[36] and PCL[37] libraries.

The MPC controller is implemented with a sampling rate $T_c = 40$ ms, considering a 6-s prediction horizon divided into $N = 6$ steps. An active-set method implementation (namely qpOases[38]) is used to solve the QP optimization. The problem consists of 56 variables ($n\,N = 48$ control variables plus $n = 8$ artificial set-point), with 56 boxed constraints and up to 457 linear constraints of the form $\mathbf{G}\mathbf{u} \leq \mathbf{b}$ (they are actually enforced on $\boldsymbol{\delta}\mathbf{U}$). During experiments, the required time to solve online the MPC problem was less than 1 ms on average, with maximum values of about 10 ms.

The inverse kinematics problem has been solved by exploiting the redundancy parameters described in Section 4 and in the appendix. Note that the youBot arm has 5 DoF, therefore the use of ρ_3 is necessary only when the desired end-effector pose is “vertical” (i.e., $\chi_{ee}^{d} = \pi$).

6.2. *Experiments*

Two experiments are reported (also refer to the accompanying video). First, the arm obstacle avoidance is tested. The robot must reach a time invariant pose $\mathbf{P}_d$. A bridge-like obstacle is placed between the starting position and the desired position. The obstacle position is not a priori known and it is detected by the Xtion mounted on the front of the platform (color segmentation is used to detect the obstacle). Figures 6 and 7 report how the arm bends backwards before an impact occurs, thanks to the enforced constraints. Figure 8 shows the behavior of the constraint enforced between the last link of the manipulator and the closest obstacle point on the first step of the prediction horizon: The upper bound limit decreases as the robot moves closer to the obstacle, until the constraint becomes

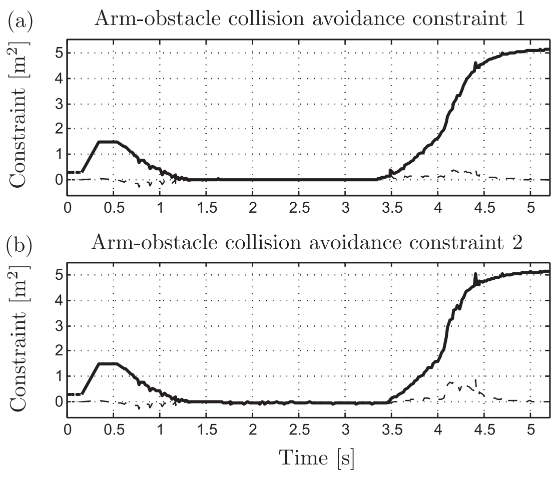


Fig. 8. Obstacle avoidance constraint for the end-effector with respect to the *obstacle* closest point on the first prediction step (first experiment). (a) First constraint of Eq. (24) with respect to the obstacle closest point. (b) Second constraint of Eq. (24). The upper bound limit is in solid black, while the left hand side of Eq. (24) in in dashed black.

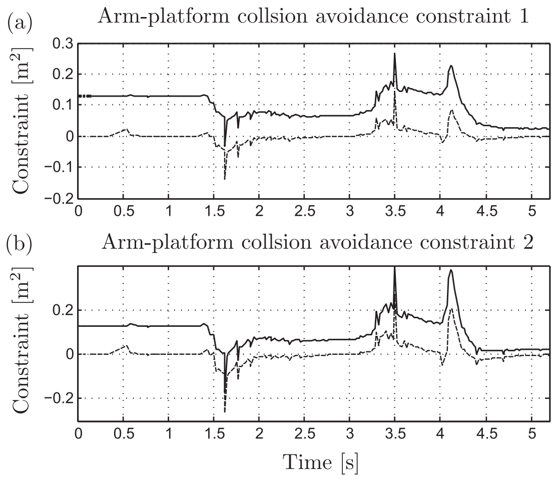


Fig. 9. Obstacle avoidance constraint for the end-effector with respect to the *platform* closest point on the first prediction step (first experiment). (a) First constraint of Eq. (24). (b) Second constraint of Eq. (24). The upper bound limit is in solid black, while the left hand side of Eq. (24) in in dashed black.

active. Figure 9 shows the collision avoidance constraint enforced between the end-effector link and the closest point on the robot platform on the first prediction step, that assures that while bending backward the arm does not collide with the platform.

The second experiment (Fig. 10) has been used to assess the controller behavior in a dynamic scenario. The task sequence is activated by a gesture of the human (hand waving) detected trough the Xtion, that asks the robot for an object. The robot picks up the object, detected through the eye-in-hand camera, and brings it to the human, avoiding collisions with obstacles (also "U-shaped"), which are unknown and detected only by the distance sensors. The FoV constraint assures that the human hand remains in the Xtion FoV. During this phase, the human hand itself is the time-varying goal position for the end-effector. To avoid collisions, during navigation the arm is kept retracted over the mobile base, and it starts reaching out when close to the goal. The necessary references for the arm are managed by the task scheduler of Fig. 5, together with the different experiment phases (looking for the object, moving the end-effector in the grasping position, moving toward the human hand, and so on). In order to handle the limited sensing range of the Xtion, the task scheduler stops the robot motion whenever the human hand is too far away. The motion is resumed when the hand is detected again. Figure 11 illustrates the x–y projection of the positions of the base, the end-effector,

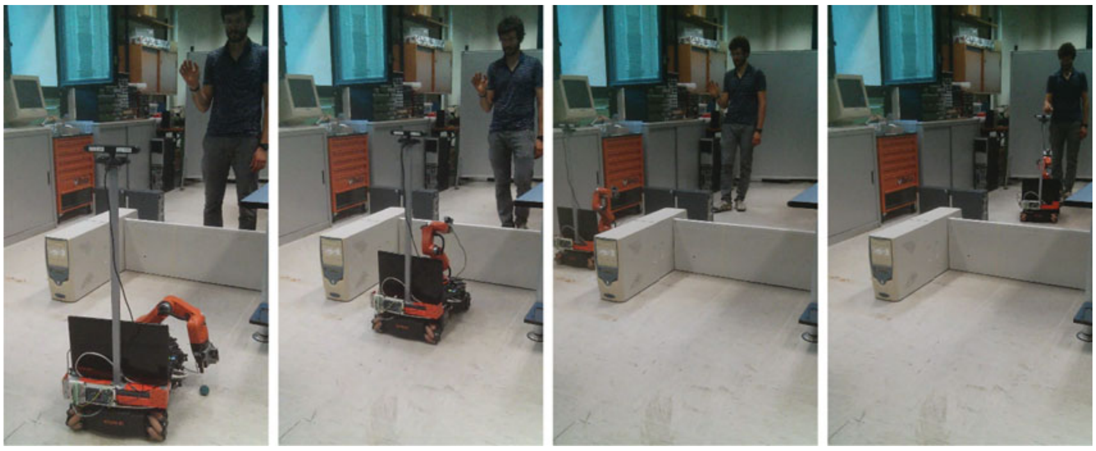


Fig. 10. Snapshots of task execution in a dynamic environment.

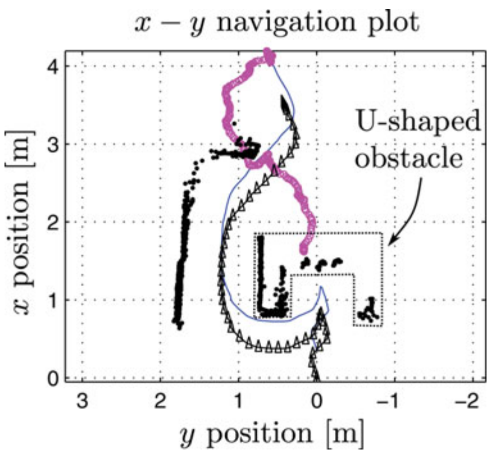


Fig. 11. x–y positions of the mobile base center (solid black and oriented black triangles), of the end-effector (blue), of the detected obstacles (black dots) and of the tracked human hand (i.e., the goal position, magenta circles) during second experiment.

the detected obstacles and the goal (i.e., the tracked hand) during the navigation phase. Figure 12(a) reports the collision avoidance constraint enforced between the platform and the obstacles detected by one of the front distance sensors, while Fig. 12(b) shows the FoV constraint.

6.3. Comments

The presented experiments show the applicability of the proposed MPC approach to mobile manipulation. The implemented control strategy is able to handle both joint space and Cartesian space time varying references during the same experiment, thanks to the inverse kinematics procedure based on the introduced redundancy parameters. The robot moves in a completely unknown environment, avoiding collisions with obstacles perceived on the fly, thanks to the imposed constraints and to the evasive motion reference that has been included in the MPC cost function. Cooperation with a human operator has been demonstrated, thus validating the viability of our approach in dynamic scenarios.

7. Conclusions

In this paper, a controller based on constrained optimization has been developed for tracking problems in mobile manipulation: An MPC problem treating the mobile manipulator as a single system has been defined in a suited velocity form that guarantees offset-free tracking, recursive feasibility, convergence and stability. The problem is set and solved online, allowing to deal with dynamic scenarios and unforeseen events. Navigation performance of the platform has been improved by including an

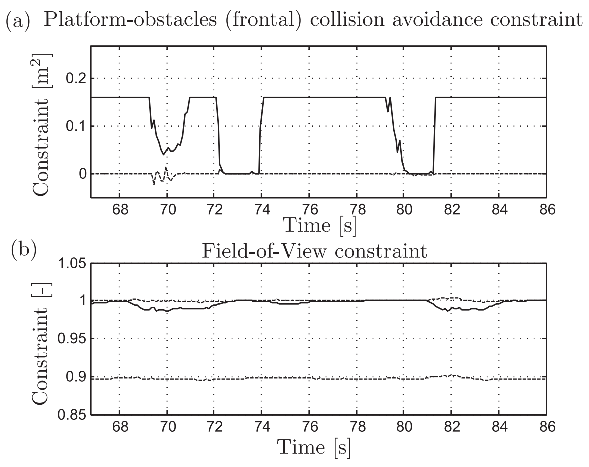


Fig. 12. (a) Obstacle avoidance constraint of Eq. (23) for the base: upper bound limit in solid black, constraint value in dashed black. (b) Field-of-view constraint of Eq. (26): upper and lower bounds in solid black, the constraint in dashed black.

additional goal to the MPC problem, derived from the vortex field approach, and implemented through an algorithm that deals with online-only perception of the environment. Tracking of Cartesian space references has also been considered, also proposing a redundancy resolution method for mobile manipulators endowed with anthropomorphic arm. Besides acceleration, velocity and position constraints, collision avoidance constraints for the mobile base and the arm and FoV constraints have been enforced and extended over the prediction horizon, while maintaining the linear-quadratic formulation of the problem. Experimental validation on a KUKA youBot mobile manipulator shows the online applicability of the presented approach.

Appendix

This appendix deals with the inverse kinematics of a mobile manipulator endowed with a six DoF anthropomorphic arm with spherical wrist ($n = 9$ DoF). Since $\mathbf{P}^d \in \mathbb{R}^6$, while $\mathbf{q} \in \mathbb{R}^n$, the inverse kinematics problem is under-constrained if $n > 6$: A certain end-effector pose can be obtained with any posture of the mobile platform of the robot $\mathbf{q}_b$, provided that it resides in the reachable workspace of the arm. To solve the inverse kinematics, a set of redundancy parameters $\boldsymbol{\rho} = [\rho_1, \rho_2, \rho_3]$ (Fig. 13) is here introduced that fully constrain $\mathbf{q}_b$, extending the results of,[39] that treated the particular case of a 5-DoF arm mounted on a mobile platform. Parameter ρ_1 describes the angular displacement between the mobile platform main axis and the x–y projection of $(\mathbf{r}_{ee} - \mathbf{r}_a)$ linking the arm base position $\mathbf{r}_a$ to the end-effector position $\mathbf{r}_{ee}$:

$$\rho_1 = \text{atan}\left((y_{ee} - y_a)/(x_{ee} - x_a)\right) - \theta_b. \tag{27}$$

Parameter ρ_2 is the length of the x–y planar projection of vector $(\mathbf{r}_{ee} - \mathbf{r}_a)$, i.e., the robotic arm extension:

$$\rho_2 = \sqrt{(x_{ee} - x_a)^2 + (y_{ee} - y_a)^2}, \tag{28}$$

ρ_3 defines the angular displacement between the end-effector orientation angle ψ_{ee} and the x–y projection of $(\mathbf{r}_{ee} - \mathbf{r}_a)$:

$$\rho_3 = \tan^{-1}\left(\frac{y_{ee} - y_a}{x_{ee} - x_a}\right) - \psi_{ee}. \tag{29}$$

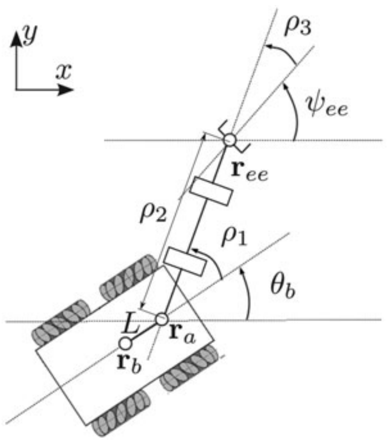


Fig. 13. Generalized mobile manipulator redundancy parameters.

Given a desired end-effector pose $\mathbf{P}^d_{ee}$, once suited parameters $\boldsymbol{\rho}^d$ are chosen, the base configuration $\mathbf{q}^d_b$ is completely defined through the following relation:

$$\mathbf{q}^d_b = \begin{bmatrix} x^d_b \\ y^d_b \\ \theta^d_b \end{bmatrix} = \begin{bmatrix} x^d_{ee} - \rho^d_2\cos(\rho^d_3 + \psi^d_{ee}) - L\cos(\rho^d_3 + \psi^d_{ee} - \rho^d_1) \\ y^d_{ee} - \rho^d_2\sin(\rho^d_3 + \psi^d_{ee}) - L\sin(\rho^d_3 + \psi^d_{ee} - \rho^d_1) \\ \rho^d_3 + \psi^d_{ee} - \rho^d_1 \end{bmatrix}$$

where L is the x–y planar projection of $\mathbf{r}_b - \mathbf{r}_a$. The standard arm inverse kinematics can then be solved, obtaining the joint values $\mathbf{q}^d_a$. First, a value ρ^d_3 should be chosen for parameter ρ_3, which can be used to reduce the mobile platform motion in the "approach phase," that is usually the first part of a mobile manipulation task. Note that, for a given value of ρ^d_2, $\mathbf{P}^d_{ee}$ can be realized with the mobile platform lying in any position that satisfies $\rho_2 = \rho^d_2$, i.e., the platform center can lie in any point of a circle centred in $[x^d_{ee}, y^d_{ee}]$, with different values of ρ_3. ρ^d_3 should be chosen to minimize the platform movement with respect to its current position. In particular, $\mathbf{r}_{a,k} = [x_{a,k}, y_{a,k}, z_{a,k}]^{\mathrm{T}}$ should always be aligned with the goal position $\mathbf{r}^d_{ee}$. In this way, during the approach phase, it is sufficient for the robot to straightforwardly move toward $\mathbf{r}^d_{ee}$. As soon as $\mathbf{r}^d_{ee}$ enters the arm reachable workspace, the positioning task can be executed, without further motions of the mobile platform to align the end-effector with the desired orientation $\boldsymbol{\epsilon}_{ee}$. Given $\mathbf{P}^d_{ee}$, and without needing to know ρ^2_d, this behavior can be achieved by setting ρ^d_3 as

$$\rho^d_3 = \mathrm{atan}\left((y^d_{ee} - y_{a,k})/(x^d_{ee} - x_{a,k})\right) - \psi^d_{ee}. \tag{30}$$

Parameter ρ_1 can be used, e.g., to maintain a certain position $\mathbf{r}_f = [x_f, y_f, z_f]^{\mathrm{T}}$ in the "platform field-of-view," which can be useful if an object placed in $\mathbf{r}_f$ has to be tracked by some platform-mounted sensing device (see Figs. 13 and 4). This can be achieved, e.g., by setting

$$\rho^d_1 = \psi^d_{ee} + \rho^d_3 - \mathrm{atan}\left((y_f - y_{a,k})/(x_f - x_{a,k})\right), \tag{31}$$

so that $\mathbf{r}_f$ planar projection always lies on the platform axis.

Finally, parameter ρ_2 has been exploited to maximize the manipulability measure[40] for the whole mobile manipulator, following the example of ref. [19]. The manipulability index $U_m(\mathbf{q})$ is defined as

$$U_m(\mathbf{q}) = \sqrt{(\partial fk(\mathbf{q})/\partial\mathbf{q})^{\mathrm{T}}(\partial fk(\mathbf{q})/\partial\mathbf{q})}. \tag{32}$$

The optimization problem has been solved with gradient descent on-line method,[41] which allows to constantly optimize the robot configuration. At each time instant k, a value $\rho^+_{2,k} = \rho_{2,k} + \gamma$ is defined, where $\gamma > 0$ is a parameter tuning the speed of convergence. Then, the corresponding values

$\mathbf{q}^+ = ik(\mathbf{P}_k, [\rho_{1,k}, \rho_2^+, \rho_{3,k}]^{\mathrm{T}})$ is computed. The value of ρ_2^d is then determined as

$$\rho_2^d = \rho_{2,k} + \gamma \cdot (U_m(\mathbf{q}^+) - U_m(\mathbf{q}_k)). \tag{33}$$

Supplementary material

For supplementary material for this article, please visit https://doi.10.1017/S0263574717000133

References

1. W. Decré, H. Bruyninckx and J. De Schutter, "Extending the Itasc Constraint-Based Robot Task Specification Framework to Time-Independent Trajectories and User-Configurable Task Horizons," *Proceedings of the IEEE International Conference on Robotics and Automation, ICRA* (May 2013) pp. 1941–1948.
2. M. de Lasa and A. Hertzmann, "Prioritized Optimization for Task-Space Control," *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS* (Oct. 2009) pp. 5755–5762.
3. A. M. Zanchettin and P. Rocco, "Near Time-Optimal and Sensor-Based Motion Planning for Robotic Manipulators," *Proceedings of the IEEE 52nd Annual Conference on Decision and Control, CDC* (Dec. 2013) pp. 965–970.
4. C. Samson, B. Espiau and M. L. Borgne, *Robot Control: The Task Function Approach* (Oxford University Press, 1991).
5. A. Del Prete, F. Romano, L. Natale, G. Metta, G. Sandini and F. Nori, "Prioritized Optimal Control," *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, May 2014, pp. 2540–2545.
6. E. Camacho and C. Bordons, *Model Predictive Control* (Springer, 2004).
7. D. Mayne, J. Rawlings, C. Rao and P. Scokaert, "Constrained model predictive control: Stability and optimality," *Automatica* **36**(6), 789–814 (2000).
8. M. Vukov, A. Domahidi, H. Ferreau, M. Morari and M. Diehl, "Auto-Generated Algorithms for Nonlinear Model Predictive Control on Long and on Short Horizons," *Proceedings of the IEEE 52nd Annual Conference on Decision and Control (CDC)* (Dec. 2013) pp. 5113–5118.
9. S. Maniatopoulos, D. Panagou and K. Kyriakopoulos, "Model Predictive Control for the Navigation of a Nonholonomic Vehicle with Field-of-View Constraints," *Proceedings of the American Control Conference (ACC)* (Jun. 2013) pp. 3967–3972.
10. T. P. Nascimento, A. P. Moreira and A. G. S. Conceição, "Multi-robot nonlinear model predictive formation control: Moving target and target absence," *Robot. Auton. Syst.* **61**(12), 1502–1515 (2013).
11. T. P. Nascimento, A. G. S. Conceição and A. P. Moreira, "Multi-robot nonlinear model predictive formation control: the obstacle avoidance problem," *Robotica* **34**, 549–567 (Mar. 2016).
12. A. Brooks, T. Kaupp and A. Makarenko, "Randomised MPC-Based Motion-Planning for Mobile Robot Obstacle Avoidance," *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)* (May 2009) pp. 3962–3967.
13. J. C. L. Barreto Sobrinho, A. Conceição, C. Dorea, L. Martinez and E. De Pieri, "Design and implementation of model-predictive control with friction compensation on an omnidirectional mobile robot," *IEEE/ASME Trans. Mechatronics* **19**(2), 467–476 (Apr. 2014).
14. M. Fruchard, P. Morin and C. Samson, "A framework for the control of nonholonomic mobile manipulators," *Int. J. Robot. Res.* **25**(8), 745–780 (2006).
15. V. Padois, J.-Y. Fourquet and P. Chiron, "Kinematic and dynamic model-based control of wheeled mobile manipulators: A unified framework for reactive approaches," *Robotica* **25**(2), 157–173 (2007).
16. P. Falco and C. Natale, "Low-level flexible planning for mobile manipulators: A distributed perception approach," *Adv. Robot.* **28**(21), 1431–1444 (2014).
17. S. Heshmati-alamdari, G. C. Karras, A. Eqtami and K. J. Kyriakopoulos, "A Robust Self Triggered Image Based Visual Servoing Model Predictive Control Scheme for Small Autonomous Robots," *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS* (Sep. 2015) pp. 5492–5497.
18. S. Ide, T. Takubo, K. Ohara, Y. Mae and T. Arai, "Real-Time Trajectory Planning for Mobile Manipulator using Model Predictive Control with Constraints," *Proceedings of the 8th International Conference on Ubiquitous Robots and Ambient Intelligence, URAI* (Nov. 2011) pp. 244–249.
19. G. Buizza Avanzini, A. M. Zanchettin and P. Rocco, "Reactive Constrained Model Predictive Control for Redundant Mobile Manipulators," *Proceedings of the 13th International Conference Intelligent Autonomous Systems, IAS-13* (Springer International Publishing 2016) pp. 1301–1314.
20. G. Buizza Avanzini, A. M. Zanchettin and P. Rocco, "Constraint-Based Model Predictive Control for Holonomic Mobile Manipulators," *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS* (2015).

21. L. Wang, “A tutorial on model predictive control: Using a linear velocity-form model,” *Dev. Chem. Eng. Miner. Process.* **12**(5–6), pp. 573–614 (2004).
22. D. Limon, I. Alvarado, T. Alamo and E. F. Camacho, “MPC for tracking piecewise constant references for constrained linear systems,” *Automatica* **44**(9), 2382–2387 (2008).
23. G. Betti, M. Farina and R. Scattolini, “A robust MPC algorithm for offset-free tracking of constant reference signals,” *IEEE Trans. Autom. Control* **58**(9), 2394–2400 (2013).
24. R. Bischoff, U. Huggenberger and E. Prassler, “KUKA Youbot - A Mobile Manipulator for Research and Education,” *Proceedings of the IEEE International Conference on Robotics and Automation, ICRA* (2011).
25. I. Alvarado, D. Limon, T. Alamo and E. Camacho, “Output Feedback Robust Tube Based MPC for Tracking of Piece-Wise Constant References,” *Proceedings of the 46th IEEE Conference on Decision and Control* (Dec. 2007) pp. 2175–2180.
26. A. Ferramosca, D. Limon, I. Alvarado, T. Alamo and E. Camacho, “MPC for tracking with optimal closed-loop performance,” *Automatica* **45**(8), 1975–1978 (2009).
27. D. Limon, I. Alvarado, T. Alamo and E. Camacho, “Robust tube-based MPC for tracking of constrained linear systems with additive disturbances,” *J. Process Control* **20**(3), 248–260 (2010).
28. C. De Medio, F. Nicol and G. Oriolo, “Robot Motion Planning Using Vortex Fields,” *New Trends in Systems Theory*, Progress in Systems and Control Theory, vol. 7 (Birkhuser, Boston, 1991) pp. 237–244.
29. B. Lacevic and P. Rocco, “Kinetostatic Danger Field - A Novel Safety Assessment for Human-Robot Interaction,” *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS* (2010).
30. A. M. Zanchettin and P. Rocco, “A general user-oriented framework for holonomic redundancy resolution in robotic manipulators using task augmentation,” *IEEE Trans. Robot.* **28**(2), 514–521 (2012).
31. N. M. Ceriani, A. M. Zanchettin, P. Rocco, A. Stolt and A. Robertsson, “Reactive task adaptation based on hierarchical constraints classification for safe industrial robots,” *IEEE/ASME Trans. Mechatron.* **20**(6), 2935–2949 (2015).
32. L. Sciavicco and B. Siciliano, “A solution algorithm to the inverse kinematic problem for redundant manipulators,” *IEEE J. Robot. Autom.* **4**(4), 403–410 (1988).
33. A. M. Zanchettin, N. M. Ceriani, P. Rocco, H. Ding and B. Matthias, “Safety in human-robot collaborative manufacturing environments: Metrics and control,” *IEEE Trans. Autom. Sci. Eng.* **13**(2), 882–893 (2016).
34. M. Ragaglia, A. M. Zanchettin, and P. Rocco, “Safety-Aware Trajectory Scaling for Human-Robot Collaboration with Prediction of Human Occupancy,” *Proceedings of the International Conference on Advanced Robotics, ICAR* (2015).
35. Open cv. [Online]. Available: `http://opencv.org/`
36. Open ni. [Online]. Available: `http://github.com/OpenNI/OpenNI`
37. Point cloud library pcl. [Online]. Available: `http://pointclouds.org/`
38. H. Ferreau, H. Bock and M. Diehl, “An online active set strategy to overcome the limitations of explicit MPC,” *Int. J. Robust Nonlinear Control* **18**(8), 816–830(2008).
39. S. Sharma, G. K. Kraetzschmar, C. Scheurer and R. Bischoff, “Unified Closed Form Inverse Kinematics for the KUKA Youbot,” *Proceedings of the ROBOTIK/German Conference on Robotics* (2012).
40. N. Vahrenkamp, T. Asfour, G. Metta, G. Sandini and R. Dillmann, “Manipulability Nnalysis,” *Proceedings of the 2012 12th IEEE-RAS International Conference on Humanoid Robots, Humanoids* (Nov. 2012) pp. 568–573.
41. J. Nocedal and S. J. Wright, *Numerical Optimization*, Springer Series in Operations Research and Financial Engineering (Springer, 2006).