Robotics: Science and Systems 2019
Freiburg im Breisgau, June 22-26, 2019

Semi-Autonomous Robot Teleoperation with Obstacle Avoidance via Model Predictive Control

Matteo Rubagotti*,Tasbolat Taunyazov†,Bukeikhan Omarali‡ and Almas Shintemirov§
*§Dept. of Robotics and Mechatronics, Nazarbayev University, Nur-Sultan (Astana), Kazakhstan
†Dept. of Computer Science, National University of Singapore, Singapore
‡School of Elect. Eng. and Computer Science, Queen Mary, University of London, London, UK.
Email: *matteo.rubagotti@nu.edu.kz, †taunyazov@gmail.com, ‡bukeikhan.omarali@gmail.com §ashintemirov@nu.edu.kz

***Abstract*—This paper [1] proposes a model predictive control approach for semi-autonomous teleoperation of robot manipulators: the focus is on avoiding obstacles with the whole robot frame, while exploiting predictions of the operator's motion. The hand pose of the human operator provides the reference for the end effector, and the robot motion is continuously replanned in real time, satisfying several constraints. An experimental case study is described regarding the design and testing of the described framework on a UR5 manipulator: the experimental results confirm the suitability of the proposed method for semi-autonomous teleoperation, both in terms of performance (tracking capability and constraint satisfaction) and computational complexity (the control law is calculated well within the sampling interval).**

I. Introduction

Teleoperation systems for robot manipulators [1]–[3] have been applied to a variety of domains ranging from space robotics [4], to robotic surgery [5], to handling of hazardous (e.g., radioactive) materials [6]. In some cases, teleoperation schemes need to be adapted to the environment, to the operator, or to the specific task [7]. Thus, *semi-autonomous teleoperation* strategies have been developed, in which the operator only focuses on providing the reference for the end-effector pose (e.g., through a joystick, a motion capture system or a brain-machine interface), and the control scheme generates a corresponding adaptive robot motion [7].

When using *model predictive control* (MPC) [8] for controlling a robot, its motion is optimally replanned at each sampling time, by defining a sequence of control moves within a given *prediction horizon* via numerical optimization. Also, thanks to the availability of ad-hoc toolboxes, such as ACADO [9], CVXGEN [10], and PANOC [11], MPC is becoming a viable solution for the control of robot manipulators (see, e.g., the recent results in [12]–[15]). When applied to teleoperation, MPC relies on a forecast of the operator's motion in the prediction horizon, which constitutes the main difference with respect to standard trajectory tracking problems, where the reference is known from the beginning.

In recent years, MPC has been applied to semi-autonomous robot teleoperation, generating constrained robot motions in the presence of obstacles. In [16], [17], obstacle shapes are not directly accounted for, but constraints are imposed on maximum contact forces, so that the robot is allowed to move in a clutter: this does not necessitate the use of long prediction horizons, which are kept at relatively short values, between 10 ms and 160 ms in the shown experimental results. In the case when contacts with obstacles have to be avoided by planning an alternative trajectory, relatively long prediction horizons would allow the robot to be proactive, exploiting the prediction of the human motion. Methods based on nonlinear MPC for obstacle avoidance with known object geometries and locations have been recently proposed outside the field of teleoperation in [18]–[22].

In this work, we propose a semi-autonomous teleoperation framework based on MPC for tracking the end-effector position and orientation of a robot manipulator with a relatively long prediction horizon (in the considered case study, of 1 s), imposing limits on joint angles, speeds and accelerations, avoiding singular configurations and collisions both between links and with obstacles. A nonlinear MPC problem is formulated to continuously replan the robot motion based on a prediction of the human hand pose available from a motion capture system, cast into a nonlinear program, and solved via *sequential quadratic programming* (SQP). Experimental results are reported for a UR5 6-DOF industrial robot-manipulator with a CB2 controller, equipped with a Robotiq 3 finger adaptive gripper, and teleoperated by an operator wearing a custom-made 7-DOF human arm motion tracker.

The combination of the above-mentioned characteristics in a single MPC framework and their experimental testing is a first contribution of this paper, which, at least to the best of the authors' knowledge, constitutes a novelty with respect to the cited works, i.e. the formulation of constraints on obstacle avoidance allows their inclusion into the MPC problem. A second contribution of the paper is the overall experimental implementation of the proposed method, showing that the computation time (critical for MPC) is largely contained in the sampling interval using state-of-the-art hardware.

Notation: Given a vector $\boldsymbol{g}$ or matrix $\boldsymbol{G}$, $\boldsymbol{g}'$ and $\boldsymbol{G}'$ are, respectively, their transpose. Given a square matrix $\boldsymbol{M} \in \mathbb{R}^{n\times n}$, the expressions $\boldsymbol{M} \succeq 0$ and $\boldsymbol{M} \succ 0$ impose that the matrix be positive semi-definite and positive definite, respectively. Given $\boldsymbol{g} \in \mathbb{R}^n$ and $\boldsymbol{M} \in \mathbb{R}^{n\times n}$, then $\|\boldsymbol{g}\|^2_{\boldsymbol{M}} \triangleq \boldsymbol{g}'\boldsymbol{M}\boldsymbol{g}$. A sequence of

[1]This paper has been selected to appear also in IEEE Robotics and Automation Letters, under the same title.

This work was funded under the Nazarbayev University faculty development grant project "Development of an Intelligent Assistive Robot Manipulation System for Improving the Quality of Life of Disabled People in Kazakhstan" (Project PI: A. Shintemirov)

integer numbers from ν_1 to ν_2 included is denoted as $\mathbb{N}_{[\nu_1,\nu_2]}$: for instance, $\mathbb{N}_{[0,3]} = \{0,1,2,3\}$.

II. MPC-based Teleoperation Framework

A. Prediction of the human hand pose

Current and predicted human hand poses are necessary to generate references for the MPC controller. Since the robot does not necessarily have an anthropomorphic structure, the configuration of the whole human arm is not directly considered as useful information: only the human hand pose relative to the human shoulder joint is taken as reference for the robot end effector. The human shoulder frame is appropriately adjusted and scaled depending on the robot dimensions and translated to the fixed robot reference frame $\bar{O}-\bar{x}\bar{y}\bar{z}$. Two pieces of information need to be generated from motion capture data: the reference position of the end effector, referred to as $\bar{\boldsymbol{p}} \in \mathbb{R}^3$, and its reference orientation, which is here defined using unit quaternions, and referred to as $\bar{\boldsymbol{q}} \in \mathbb{R}^4$. The overall human hand pose is thus referred to as

$$\bar{\boldsymbol{x}} = \begin{bmatrix} \bar{\boldsymbol{p}} \\ \bar{\boldsymbol{q}} \end{bmatrix} \in \mathbb{R}^7. \tag{1}$$

Given the current time instant t_0, in which $\bar{\boldsymbol{x}}$ is acquired from the motion capture system, the human hand pose forecast along the prediction horizon, i.e., for time t in the interval

$$\mathcal{T} \triangleq [t_0, t_0 + T_p], \tag{2}$$

is referred to as

$$\hat{\boldsymbol{x}}_{\mathcal{T}} = \begin{bmatrix} \hat{\boldsymbol{p}}_{\mathcal{T}} \\ \hat{\boldsymbol{q}}_{\mathcal{T}} \end{bmatrix}, \tag{3}$$

where $\bar{\boldsymbol{x}} = \hat{\boldsymbol{x}}(t_0)$. Predictions can be obtained depending on the specific case study (see, e.g., [23]–[25]).

B. Robot kinematics and dynamics for MPC

1) Forward kinematics: While the robot motion is described in the joint space, tracking the reference $\bar{\boldsymbol{x}}$ defined in (1) requires using the task space. Furthermore, in order to avoid obstacles with the whole robot frame, the most convenient way is to impose that a number of *test points* in the robot frame are kept sufficiently far from the obstacles. Therefore, forward kinematics is needed in order to express the robot end effector pose $\boldsymbol{x}_e$ and the test point positions in the task space, starting from the vector of joint variables $\boldsymbol{\theta} \in \mathbb{R}^{n_\theta}$. Specifically, we obtain the end-effector orientation by using unit quaternions. Vector $\boldsymbol{x}_e(\boldsymbol{\theta})$ can be split into position components $\boldsymbol{p}_e(\boldsymbol{\theta}) \in \mathbb{R}^3$ and orientation component (via unit quaternions) $\boldsymbol{q}_e(\boldsymbol{\theta}) \in \mathbb{R}^4$, as

$$\boldsymbol{x}_e = \begin{bmatrix} \boldsymbol{p}_e \\ \boldsymbol{q}_e \end{bmatrix} \in \mathbb{R}^7. \tag{4}$$

Regarding the test points, only their positions are important for obstacle avoidance purposes. These positions are determined by using a part of the calculations necessary to obtain $\boldsymbol{p}_e$, depending on their position along the robot frame. The choice of their locations is tightly linked to the obstacle avoidance problem, and will be therefore described in Section II-C.

2) Singularity-free motion: The avoidance of singular configurations can be obtained by imposing, in general, that

$$\omega(\boldsymbol{\theta}) = \sqrt{\det(\boldsymbol{J}(\boldsymbol{\theta})\boldsymbol{J}'(\boldsymbol{\theta}))} \geq \tilde{\omega} > 0, \tag{5}$$

where $\omega(\boldsymbol{\theta})$ is the volume of the manipulability ellipsoid, $\boldsymbol{J}(\boldsymbol{\theta})$ is the geometric Jacobian of the robot, and $\tilde{\omega}$ is the imposed minimum manipulability measure [26]. Not going through singular configurations is important to reduce the occurrence of sudden peaks of joint accelerations and/or velocities.

3) Dynamic models: The proposed scheme uses a state-space model to generate predictions of the robot motion. In general, the model is defined in the joint space, as

$$\dot{\boldsymbol{z}} = f(\boldsymbol{z}, \boldsymbol{u}) \tag{6}$$

where $\boldsymbol{z} \in \mathbb{R}^{n_z}$ and $\boldsymbol{u} \in \mathbb{R}^{n_u}$ are the state and input vectors, respectively. Different choices are possible for defining (6), but in practice it is often convenient to provide joint position and/or velocity references to the internal control loops of the manipulator. By using all the joint positions and velocities as states, one can write (6) as

$$\dot{\boldsymbol{z}} = \boldsymbol{A}\boldsymbol{z} + \boldsymbol{B}\boldsymbol{u} \tag{7}$$

where the state vector is $\boldsymbol{z} = \begin{bmatrix} \boldsymbol{\theta}' & \dot{\boldsymbol{\theta}}' \end{bmatrix}' \in \mathbb{R}^{n_z}$, with $n_z = 2n_\theta$, while the control vector $\boldsymbol{u} \in \mathbb{R}^{n_u}$, with $n_u = n_\theta$, represents the joint accelerations. This implies that

$$\boldsymbol{A} = \begin{bmatrix} \boldsymbol{0} & \boldsymbol{I} \\ \boldsymbol{0} & \boldsymbol{0} \end{bmatrix} \in \mathbb{R}^{n_z \times n_z}, \ \boldsymbol{B} = \begin{bmatrix} \boldsymbol{0} \\ \boldsymbol{I} \end{bmatrix} \in \mathbb{R}^{n_z \times n_u},$$

where $\boldsymbol{0} \in \mathbb{R}^{n_u \times n_u}$ and $\boldsymbol{I} \in \mathbb{R}^{n_u \times n_u}$ are zero and identity matrices, respectively.

C. Obstacle avoidance constraints

In the following, we consider two types of obstacle shapes. The first type describes half-space constraints, typically used when floors, ceilings, or walls (referred to as *Type-1 obstacles*) are in the robot workspace. For an arbitrary number n_h of Type-1 obstacles, the portion of cartesian space not occupied by the obstacle is a polyhedron, and can be expressed as

$$\mathcal{P} = \{\boldsymbol{p} \ : \ \boldsymbol{A}_h\boldsymbol{p} \leq \boldsymbol{b}_h\}, \tag{8}$$

where $\boldsymbol{p}$ is the generic spatial coordinate in the robot $O-xyz$ reference frame, while $\boldsymbol{A}_h \in \mathbb{R}^{n_h \times 3}$ and $\boldsymbol{b}_h \in \mathbb{R}^{n_w}$ define the linear constraints. In order to ensure that the whole robot frame does not exit $\mathcal{P}$, spheres $\mathcal{S}_i$ of radius r_i are generated having each corresponding test point $\boldsymbol{p}_i$ as center. The values of r_i must be chosen such that, given any pair of spheres $\mathcal{S}_i$ and $\mathcal{S}_{i+1}$, their convex hull includes the whole link $\mathcal{L}_i$. For each test point, the constraints in (8) are modified, by moving each hyperplane (defined by a specific row of $\boldsymbol{A}_h$ and the corresponding value of $\boldsymbol{b}_h$) perpendicularly to itself of r_i, so as to reduce the volume of $\mathcal{P}$. The obtained polytope for each test point is denoted as

$$\mathcal{P}_i = \{\boldsymbol{p} \ : \ \boldsymbol{A}_{h,i}\boldsymbol{p} \leq \boldsymbol{b}_{h,i}\} \tag{9}$$

where $\boldsymbol{A}_{h,i}$ and $\boldsymbol{b}_{h,i}$ represent the new "tightened" values of the hyperplane positions. Due to the convexity of $\mathcal{P}_i$, one can easily obtaine that, if $\boldsymbol{p}_i \in \mathcal{P}_i$ for $i \in \mathbb{N}_{[1,n_1]}$, then the robot frame does not collide with any of the Type-1 obstacles.

The second type of shape is used for obstacles that enclose a limited portion of the robot workspace, around which the robot should be allowed to move (*Type-2 obstacles*). A number of additional test points $\tilde{\boldsymbol{p}}_i$, $i \in \mathbb{N}_{[1,n_2]}$ (in general different from points $\boldsymbol{p}_i$) are considered on the robot frame. Each point $\tilde{\boldsymbol{p}}_i$ is the center of a sphere $\tilde{\mathcal{S}}_i$, which has radius $\tilde{r}_i$, and the spheres are defined such that their union encloses the whole robot frame. Different choices can be made to model Type-2 obstacles: the method proposed in this paper consists of using ellipsoidal sets, which provides a generalization of the standard use of spherical obstacles, already used within an MPC setting in [20]. In order for the robot not to collide with the j-th obstacle, each test point $\tilde{\boldsymbol{p}}_i$ is imposed to belong to the complement of an open ellipsoidal sets, defined as

$$\bar{\mathcal{E}}_{i,j} = \left\{ \boldsymbol{p} \ : \ (\boldsymbol{p} - \boldsymbol{p}_{i,j}^c)' \boldsymbol{M}_{i,j} (\boldsymbol{p} - \boldsymbol{p}_{i,j}^c) \geq 1 \right\}, \tag{10}$$

where $\boldsymbol{p}_{i,j}^c$ is the center of the ellipsoid, matrix $\boldsymbol{M}_{i,j} \in \mathbb{R}^{3\times 3}$, with $\boldsymbol{M}_{i,j} = \boldsymbol{M}_{i,j}' \succ 0$, defines its shape, and $j \in \mathbb{N}_{[1,n_e]}$ is the index of the specific Type-2 obstacle, with n_e representing their total number. A possible method to define sets $\bar{\mathcal{E}}_{i,j}$ is proposed in the following. First, a set of vertices $\{v_{j,1}, v_{j,2}, \ldots, v_{j,n_{vj}}\}$ is defined in the task space of the robot, so that the space $\mathcal{O}_j$ occupied by the j-th obstacle satisfies $\mathcal{O}_j \subseteq \mathrm{co}(v_{j,1}, v_{j,2}, \ldots, v_{j,n_{vj}})$, n_{vj} being the number of vertices used to define the boundaries of the j-th obstacle, and $\mathrm{co}(v_{j,1}, v_{j,2}, \ldots, v_{j,n_{vj}})$ being its convex hull. Associated to each test point $\tilde{\boldsymbol{p}}_i$, regular polyhedra $\mathcal{C}_1^{i,j}, \mathcal{C}_2^{i,j} \ldots, \mathcal{C}_{n_{vj}}^{i,j}$ are constructed, each of them centered on the corresponding vertex of the j-th obstacle, and whose inscribed sphere has radius equal to $\tilde{r}_i$. The set of all vertices of all regular polyhedra associated to the vertices of the j-th obstacle and the i-th test point is referred to as $\mathcal{V}_{i,j}$. For each pair (i,j), the values of $\boldsymbol{M}_{i,j}$ and $\boldsymbol{p}_{i,j}^c$, needed to define $\bar{\mathcal{E}}_{i,j}$ in (10), can be obtained as those defining the open ellipsoid

$$\mathcal{E}_{i,j} = \left\{ \boldsymbol{p} \ : \ (\boldsymbol{p} - \boldsymbol{p}_{i,j}^c)' \boldsymbol{M}_{i,j} (\boldsymbol{p} - \boldsymbol{p}_{i,j}^c) < 1 \right\} = \mathbb{R}^3 \setminus \bar{\mathcal{E}}_{i,j}, \tag{11}$$

as the minimum-volume ellipsoid containing all points in $\mathcal{V}_{i,j}$, which can be done by solving a convex optimization problem, as detailed, e.g., in [27]. We report the following derivation, which, to the best of the authors' knowledge, is not reported in any previous works.

Proposition 1: If the robot configuration is such that $\tilde{\boldsymbol{p}}_i \in \bar{\mathcal{E}}_{i,j}$ for all $i \in \mathbb{N}_{[1,n_\theta]}$ and all $j \in \mathbb{N}_{[1,n_e]}$, then this is a collision-free configuration for Type-2 obstacles, i.e.,

$$\bigcup_{k=1}^{n_\theta} \mathcal{L}_k \cap \bigcup_{j=1}^{n_e} \mathcal{O}_j = \emptyset. \tag{12}$$

Proof: Condition (12) is implied by $\tilde{\mathcal{S}}_i \cap \mathcal{O}_j = \emptyset$, for all $i \in \mathbb{N}_{[1,n_\theta]}$ and all $j \in \mathbb{N}_{[1,n_e]}$. A new set is generated as $\mathcal{O}'_{i,j} = \mathrm{int}(\mathrm{co}(\tilde{\mathcal{C}}_1^{i,j}, \tilde{\mathcal{C}}_2^{i,j}, \ldots, \tilde{\mathcal{C}}_{n_{vj}}^{i,j}))$, where $\tilde{\mathcal{C}}_k^{i,j}$, $k \in \mathbb{N}_{[1,n_{vj}]}$, are open spheres of radius $\tilde{r}_i$, each of them centered at the corresponding vertex $v_{j,k}$ of the j-th Type-2 obstacle, while the symbol $\mathrm{int}(\cdot)$ denotes the interior of a set (i.e., $\mathcal{O}'_{i,j}$ is an open set). Given the fact that, by construction, $\tilde{\mathcal{C}}_k^{i,j} \subset \mathcal{C}_k^{i,j}$, and, by definition of $\mathcal{E}_{i,j}$, one has that $\mathcal{O}'_{i,j} \subset \mathcal{E}_{i,j}$. By construction of $\mathcal{O}'_{i,j}$, every point of its boundary has minimum Euclidean distance equal to $\tilde{r}_j$ from the boundary of $\mathcal{O}_j$. As a consequence, if $\boldsymbol{p}_i \notin \mathcal{O}'_{i,j}$, then $\tilde{\mathcal{S}}_i \cap \mathcal{O}_j = \emptyset$. But the initial assumption $\tilde{\boldsymbol{p}}_i \in \bar{\mathcal{E}}_{i,j}$ implies that $\boldsymbol{p}_i \notin \mathcal{O}'_{i,j}$, and the proposition is therefore proven. ■

For the sake of compactness, the following set is defined:

$$\bar{\mathcal{E}}_i \triangleq \bigcap_{j=1}^{n_e} \bar{\mathcal{E}}_{i,j}. \tag{13}$$

By definition, condition $\tilde{\boldsymbol{p}}_i \in \bar{\mathcal{E}}_i$ implies that the sphere $\tilde{\mathcal{S}}_i$ does not intersects with any Type-2 obstacles.

Remark 1: For both Type-1 and Type-2 obstacles, the shown results ensure that the robot frame will at most enter into contact with the obstacle boundary, but will never enter the obstacle's interior. In any practical application this would not be acceptable: the sets representing the obstacles are artificially enlarged by adding an external "safety zone", so as to ensure that the robot never moves beyond a given tolerance from the physical obstacles. This can be done by applying the described procedure using larger spheres around the test points, or considering already enlarged sets $\mathcal{O}_j$, being careful not to be too conservative, as this would imply unnecessary deviations of the robot trajectory from its reference. The presence of the safety zone makes the proposed approach unsuitable for cluttered environments, for which methods such as [16], [17] provide better solutions. □

Remark 2: From a practical standpoint, it is important to notice that a part of the robot frame could never possibly enter into collision with a given obstacle. In such cases, the corresponding set of constraints can be safely removed from the MPC problem, in order to reduce its complexity. □

Remark 3: The test points used for Type-1 and Type-2 obstacles are in general different. Indeed, at most two test points for each link are sufficient to avoid Type-1 obstacles, thanks to the convexity of the corresponding free space. Instead, a different set of test points is typically needed to avoid Type-2 obstacles, as the union of the corresponding spheres needs to cover the whole manipulator frame. □

D. Formulation of the MPC problem

Once the current state $\boldsymbol{z}_0 \triangleq \boldsymbol{z}(t_0)$ is acquired, the MPC controller will determine an optimal control signal $\boldsymbol{u}^*(t)$ in the time interval $\mathcal{T}$ defined in (2), referred to in short as $\boldsymbol{u}^*_{\mathcal{T}}$. The cost function to be minimized always accounts for the trade-off between tracking performance and moderation of control energy along the predicted robot motion. To define it, it is convenient to introduce an *output vector*, as

$$\boldsymbol{y}(\boldsymbol{z}(t)) \triangleq \left[(\boldsymbol{x}(\boldsymbol{\theta}(t)) - \hat{\boldsymbol{x}}(t))' \quad \dot{\boldsymbol{\theta}}(t)' \right]' \in \mathbb{R}^{7+n_q} \tag{14}$$

where $\boldsymbol{x}(\boldsymbol{\theta}(t)) - \hat{\boldsymbol{x}}(t)$ represents the tracking error on the end effector pose, while $\dot{\boldsymbol{\theta}}(t)$ contains the joint speeds, to

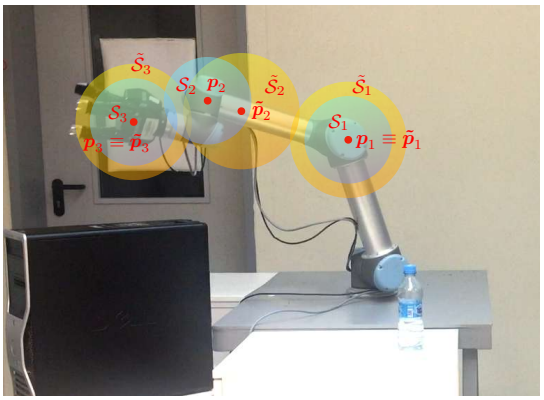

Fig. 1. Graphical representation of test points for both Type-1 and Type-2 obstacles, and corresponding spheres, for the UR5 case study.

be kept low when possible, to avoid wear of the mechanical components, reduce energy consumption, and obtain a smooth motion. The symbols $\boldsymbol{u}_{\mathcal{T}}$ and $\boldsymbol{z}_{\mathcal{T}}$ denote general realizations of $\boldsymbol{u}(t)$ and $\boldsymbol{z}(t)$, respectively, in $\mathcal{T}$. The most common choice for MPC is a quadratic function, which for the considered problem can be defined as

$$J(\boldsymbol{u}_{\mathcal{T}}, \boldsymbol{z}_{\mathcal{T}}) \triangleq \int_{t_0}^{t_0+T_p} \|\boldsymbol{y}(\boldsymbol{z}(t))\|^2_{\boldsymbol{Q}} + \|\boldsymbol{u}(t)\|^2_{\boldsymbol{R}} dt \qquad (15)$$

where $\boldsymbol{Q} = \boldsymbol{Q}' \in \mathbb{R}^{(7+n_q)\times(7+n_q)}$ and $\boldsymbol{R} = \boldsymbol{R}' \in \mathbb{R}^{n_\theta \times n_\theta}$ are chosen such that $\boldsymbol{Q} \succeq 0$, and $\boldsymbol{R} \succ 0$.

Remark 4: The presence of the tracking error $\boldsymbol{x}(\boldsymbol{\theta}(t)) - \hat{\boldsymbol{x}}(t)$ in the output vector, together with the structure of the cost function (15), would lead to minimizing the mean square prediction error on the end effector pose, in case the other terms of the cost function and the constraints were not present. This approach *does not require the use of inverse kinematics*, as the MPC controller works directly on joint variables, and minimizes a cost function depending on both the present and the predicted nonlinear output $\boldsymbol{y}(\boldsymbol{z})$, whose expression directly links joint and task spaces. □

The value of $\boldsymbol{u}^*_{\mathcal{T}}$, together with the corresponding optimal state evolution $\boldsymbol{z}^*_{\mathcal{T}}$, is obtained as solution to the following finite-horizon optimal control problem, referred to as *MPC problem* in the remainder of the paper:

$$(\boldsymbol{u}^*_{\mathcal{T}}, \boldsymbol{z}^*_{\mathcal{T}}) = \arg\min_{\boldsymbol{u}_{\mathcal{T}}, \boldsymbol{z}_{\mathcal{T}}} \; J(\boldsymbol{u}_{\mathcal{T}}, \boldsymbol{z}_{\mathcal{T}}) \qquad (16a)$$

$$\text{subj. to } \boldsymbol{z}(t_0) = \boldsymbol{z}_0 \qquad (16b)$$

$$\dot{\boldsymbol{z}}(t) = f(\boldsymbol{z}(t), \boldsymbol{u}(t)) \; t \in \mathcal{T} \qquad (16c)$$

$$\omega(\boldsymbol{z}(t)) \geq \tilde{\omega}, \; t \in \mathcal{T}, \qquad (16d)$$

$$\boldsymbol{u}(t) \in \mathcal{U}, \; t \in \mathcal{T} \qquad (16e)$$

$$\boldsymbol{z}(t) \in \mathcal{Z}, \; t \in \mathcal{T} \qquad (16f)$$

$$\boldsymbol{p}_i(\boldsymbol{z}(t)) \in \mathcal{P}_i, \; t \in \mathcal{T}, \; i \in \mathbb{N}_{[0,n_1]} \qquad (16g)$$

$$\tilde{\boldsymbol{p}}_i(\boldsymbol{z}(t)) \in \bar{\mathcal{E}}_i, \; t \in \mathcal{T}, \; i \in \mathbb{N}_{[1,n_2]}. \qquad (16h)$$

Condition (16b) sets the initial state value, while (16c) imposes that the predicted system dynamics along the prediction horizon evolves, depending on the input signal $\boldsymbol{u}(t)$, according to the model defined in (6). The inequality constraint (16d) ensures that no near-singular configurations are present in the predicted robot motion, while (16g)-(16h) impose obstacle avoidance for all the test points related to Type-1 and Type-2 obstacles, respectively. Finally, the inequality constraints (16e) and (16f) ($\mathcal{U}$ and $\mathcal{Z}$ being compact sets) are used to impose bounds on joint accelerations, and joint speeds and angles, respectively, so as to guarantee a smooth motion, while avoiding self-collisions. Following a common practice in practical MPC implementations, all inequality constraints on the states are imposed as soft constraints, while the input constraints are kept as hard constraints.

Remark 5: Due to the above-mentioned use of soft constraints, recursive feasibility (see, e.g., [8]) is guaranteed by definition. Due to slight mismatches between nominal and actual robot dynamics, and to the use of approximate solutions of the MPC problem, small violations of the soft constraints can occur in practice, for instance when nominal predictions are very close to obstacle boundaries. Regarding stability, some results are available on stabilizing MPC for reference tracking of nonlinear systems (typically for piecewise-constant references), for which the reader is referred to [28] and the references therein. These approaches typically require the introduction of additional assumptions and conditions that would increase the conservativity of the proposed approach: for this reason, no direct proof of closed-loop stability is provided, testing the performance of the MPC controller directly on the experimental setup. The investigation of MPC-based teleoperation with guaranteed a-priori closed-loop stability is beyond the scope of this contribution, and will be the object of future work. □

III. Case Study

A 7-DOF human arm motion tracker system, consisting of two IMU sensors placed on the operator's upper arm and wrist, and a potentiometer measuring the elbow angle, was designed as an improvement of work [29]. By combining the readings from these sensors, the value of $\bar{\boldsymbol{x}}$ is computed in the tracker reference frame via quaternion-based forward kinematics and translated to the base frame of the UR5 robot used in this case study with an offset in the z-axis direction equal to half the length of link 2. The MPC control scheme works with a sampling interval $T_s = 100$ ms. Since our focus is on a general MPC framework rather than on specific teleoperation scenarios, a very simple data extrapolation method is used for prediction, by generating, at every sampling time, a predicted human hand pose of ten time steps with a 100 ms spacing, with a total prediction horizon $T_p = 1$ s. The value of T_p has been chosen to provide a sufficiently long prediction of the hand pose, thus improving the performance of the MPC controller: a shorter T_p would decrease the ability of the control scheme to proactively respond to possible constraint violations; on the other hand, a longer T_p would also require to predict the human motion for a longer time, possibly leading to unreliable forecasts. On the other hand, the use of a larger T_s while keeping $T_p = 1$ s would have increased the computational

complexity of the MPC controller without a significant improvement in the tracking performance: the relatively large value for T_s is also allowed by the fact that the MPC controller provides references to internal joint controllers, which work with a faster sampling rate. The value of $\boldsymbol{p}$ is predicted via linear interpolation, which assumes a straight-line motion of the human hand in the cartesian space. The value of $\boldsymbol{q}$ is instead assumed constant in the prediction horizon. The use of a simple prediction model can be justified by the consideration that, when dealing with mixed-initiative robotic systems with human inputs such as the one considered in this letter, complicated prediction models can result in being more confusing or frustrating for the user, and thus often "the simplest prediction methods outperform the more complex ones" [24]. The forward kinematics of the UR5 robot is formulated via homogeneous transformation matrices using the Denavit-Hartenberg parameters provided by the manufacturer. Based on these parameters, the effector position $\boldsymbol{p}_e(\theta) = \begin{bmatrix} p_x & p_y & p_z \end{bmatrix}'$ and quaternion orientation $\boldsymbol{q}_e(\theta) = \begin{bmatrix} q_s & q_1 & q_2 & q_3 \end{bmatrix}'$ are obtained as nonlinear functions of the joint variables. Being the UR5 a non-redundant manipulator, it is known that equation (5) reduces to $\omega(\boldsymbol{\theta}) = |\det(\boldsymbol{J}(\boldsymbol{\theta}))| \geq \tilde{\omega}$, where, $\boldsymbol{J}(\boldsymbol{\theta})$ is obtained in the analytical form using standard methods, while $\tilde{\omega} = 10^{-9}$ has been chosen via trial and error during experiments. The UR5 robot consists of 6 links, i.e., $n_\theta = 6$ and has a pre-implemented internal controller that allows fast and precise tracking of joint velocity or position set-points by streaming the URScript language commands via TCP/IP communication from a control PC through a communication driver [30]–[32]. The same connection is used to get readings from the UR5 sensors. This allowed us to place the MPC controller on top of the internal control loop using the multivariable double integrator model described in Section II-B3, with $n_z = 2n_\theta = 12$, and $n_u = n_\theta = 6$. The manipulator is placed on a table, on top of which a computer case is also present: collisions must therefore be avoided with both objects. Maintaining the whole manipulator above the table can be expressed as a Type-1 constraint. To this end, $n_1 = 3$ test points $\boldsymbol{p}_i$ (including the end-effector position), are located on the robot frame, as shown in Fig. 1, with respect to the $O - xyz$ reference frame. All spheres $\mathcal{S}_i$ are defined with the same radius $r_i = 0.15$ m. The constraint is expressed in form (8) with $\mathcal{P} = \left\{\boldsymbol{p} \ : \ \begin{bmatrix} 0 & 0 & -1 \end{bmatrix} \boldsymbol{p} \leq 0\right\}$. Taking into account the given common value of r_i, for all $n_1 = 3$ test points we obtain the same expression for $\mathcal{P}_i$, in form (9), as

$$\mathcal{P}_i = \left\{\boldsymbol{p} \ : \ \begin{bmatrix} 0 & 0 & -1 \end{bmatrix} \boldsymbol{p} \leq -0.15\right\}, \ i = 1, 2, 3. \quad (17)$$

The computer case is instead modeled as a Type-2 obstacle. A different set of $n_2 = 3$ test points $\tilde{\boldsymbol{p}}_i$ is also represented in Fig. 1. These points and the corresponding spheres $\tilde{\mathcal{S}}_i$ are defined to avoid the collision of links 2 to 6 with the computer case (for the first link collision cannot happen, see Remark 2). All spheres $\tilde{\mathcal{S}}_i$ have the same radius $\tilde{r}_i = 0.20$ m, and $n_e = 1$ Type-2 obstacle is present. The whole set of Type-2 constraints can be formulated as in (10) and (13), as

$$\bar{\mathcal{E}}_i = \{\boldsymbol{p} \ : \ \delta_i(\boldsymbol{p}) \geq 1\}, i = 1, 2, 3, \quad (18)$$

with

$$\delta_i(\boldsymbol{p}) \triangleq \left(\boldsymbol{p} - \begin{bmatrix} 0 \\ 0.55 \\ 0 \end{bmatrix}\right)' \begin{bmatrix} 5.35 & 0 & 0 \\ 0 & 3.16 & 0 \\ 0 & 0 & 3.16 \end{bmatrix} \left(\boldsymbol{p} - \begin{bmatrix} 0 \\ 0.55 \\ 0 \end{bmatrix}\right),$$

which is the same for all $n_2 = 3$ test points, due to the fact that the three values of $\tilde{r}_i$ coincide.

For the considered application, all states are measured, and $\boldsymbol{y} \in \mathbb{R}^{13}$ is constructed as in (14). The cost function in (15), tuned via trail and error, is defined using weights in the diagonal matrix $\boldsymbol{Q}$ equal to 1 for each components of $\boldsymbol{x}(\boldsymbol{\theta}(t)) - \hat{\boldsymbol{x}}(t)$, and equal to 10^{-3} for each component of $\dot{\boldsymbol{\theta}}(t)$. Matrix $\boldsymbol{R}$ is also diagonal, with all elements on the main diagonal equal to 10^{-3}. The construction of the inequality constraints is carried out as described in Section II-C, with $n_1 = n_2 = 3$. In the MPC problem formulated as in (16), constraints (16e) and (16f) are set to impose that each component u_i of $\boldsymbol{u}$ is such that $|u_i| \leq 5$ rad/s^2, and each component $\dot{\theta}_i$ of $\dot{\boldsymbol{\theta}}$ satisfies $|\theta_i| \leq 1.5$ rad/s. Also, to avoid self-collisions, it is imposed that $\theta_4 \in [-\pi, 0]$ rad and $\theta_5 \in [-2, 2]$ rad. Constraints (16g)-(16h) are formulated with the constraint sets introduced in (17) and (18), for Type-1 and Type-2 obstacles, respectively. A direct multiple-shooting approach is applied for the discretization and solution of the MPC problem (16): T_p is divided using $N = 10$ equally-spaced nodes separated by a time step of 100 ms, which coincides with T_s. The choice of N, and consequently T_p, was dictated by the need to avoid excessive delays due to the computational complexity of the optimization problem: as previously mentioned, although a sampling interval of 100 ms is large for a robotic application, the MPC controller in practice is updating the references for low-level controllers, which act at much higher sampling rates. The resulting nonlinear program is solved via sequential quadratic programming (SQP) using the ACADO Toolkit [9]: a total of 5 SQP steps are executed at each sampling instant, and a Gauss-Newton approximation of the Hessian of the Lagrangian is used to obtain the QP approximation. Each QP is solved in condensed form using the solver qpOASES [33].

IV. Experimental results and discussion

This section presents and discusses experimental results for the considered UR5 case study. The control scheme is implemented in the Unity Game Engine programming platform (www.unity3d.com) and runs on a Lenovo notebook with Intel Core i7-860 2.3 MHZ CPU, and 8 GB RAM.

As can be seen in Fig. 2, the manipulator moves towards the bottle in Fig. 2(a), gets close to it in Fig. 2(b) and grasps it in Fig. 2(c). It moves towards the computer case in Fig. 2(d) and passes above it in Fig. 2(e). Then, it releases the bottle on the table at the opposite side of the computer case in Fig. 2(f), and moves again back towards the initial position of the bottle in Figs. 2(g) and 2(h). Finally, it moves again above the

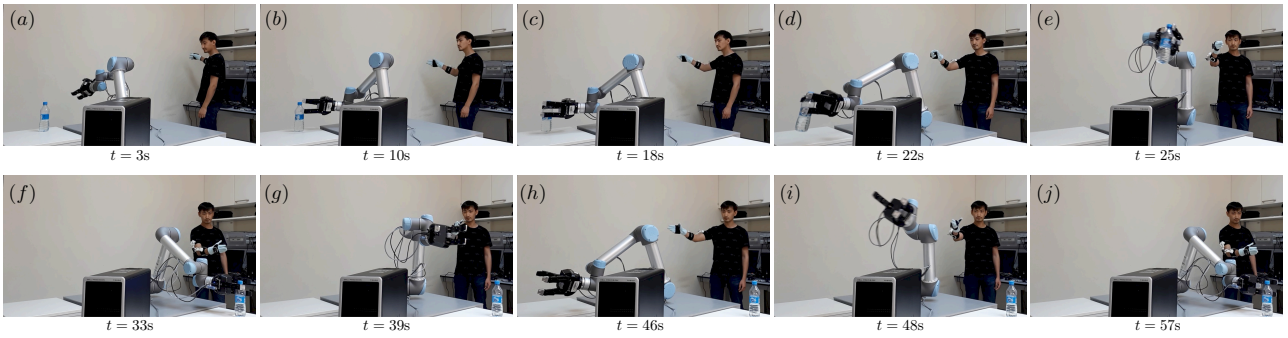

Fig. 2. Video frames at ten different time instants during the experiment. An explanation of the single subfigures is provided in the initial part of Sec. IV.

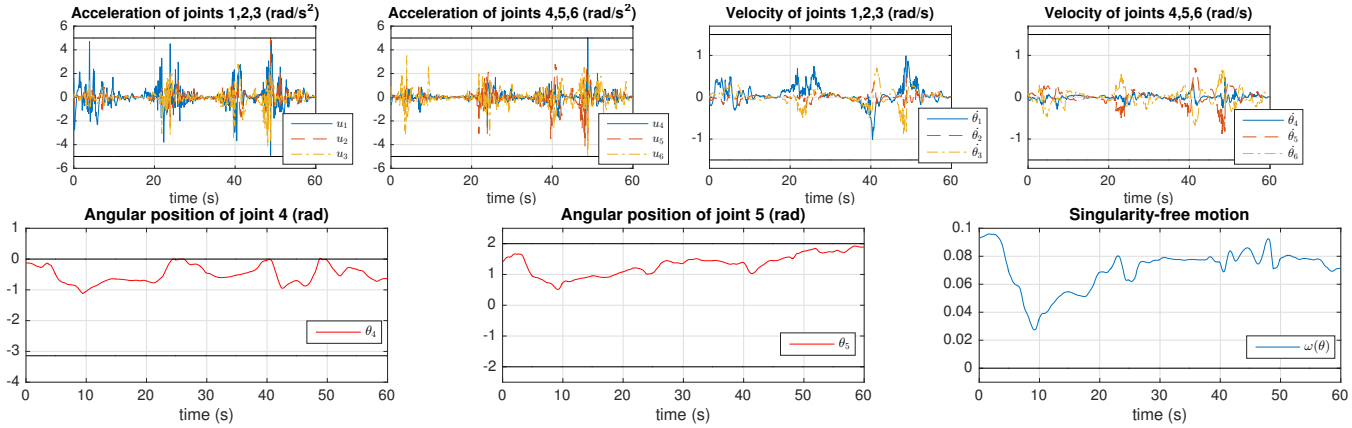

Fig. 3. Measured time evolution of, from left to right, respectively: joint accelerations u_i and joint speeds $\dot{\theta}_i$, $i = 1, \ldots, 6$, angular position θ_4, angular position θ_5, and manipulability measure $\omega(\boldsymbol{\theta})$. The corresponding boundaries are represented as solid black lines.

computer case in Fig. 2(i) to grasp the bottle a second time in Fig. 2(j).[2]

Fig. 3 shows that joint accelerations and velocities are bounded, as imposed by the MPC controller. Also, it shows that θ_4 and θ_5 are kept within the imposed boundaries so as to avoid self-collisions. Finally it displays the manipulability measure $\omega(\boldsymbol{\theta})$, which is always above the imposed threshold.

The upper plot of Fig. 4 displays the z-coordinate of each of the three test points for Type-1 obstacles (indicated in the figure as $z(\boldsymbol{p}_i)$), imposed by (17) to be above 0.15 m. The only test point that gets into contact with the constraint boundary is $\boldsymbol{p}_3$ (end effector position). The constraint is slightly violated for 0.6 s around $t = 46$ s, where $z(\boldsymbol{p}_i)$ reaches 0.145 m. This is due to the uncertainty in the prediction of the human hand pose, and to the fact that the detailed manipulator dynamics is not used in the MPC problem. Small violations of the imposed constraints are often observed in practical applications, which is the reason for defining "safety zones" (see Remark 1). The plot also shows (dotted line) the corresponding evolution $\bar{z}(\boldsymbol{p}_3)$ for the end effector in case the reference were perfectly tracked, and one can see the large constraint violation that would occur between $t = 8$ s and $t = 19$ s.

[2]The video demonstration of this work is available at http://www.alaris.kz and http://youtu.be/P9rCNJrjTjw.

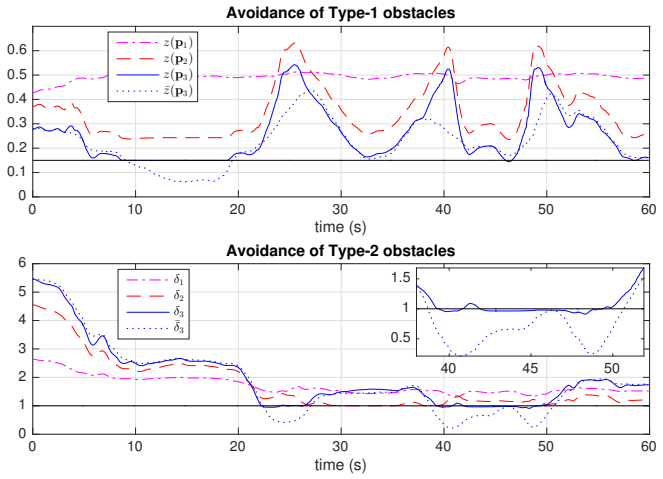

Fig. 4. Measured time evolution of, from top to bottom, respectively: z coordinates of the test points related to the avoidance of Type-1 obstacles, and value of δ_i related to the avoidance of Type-2 obstacles. The corresponding lower bounds are represented as solid black lines.

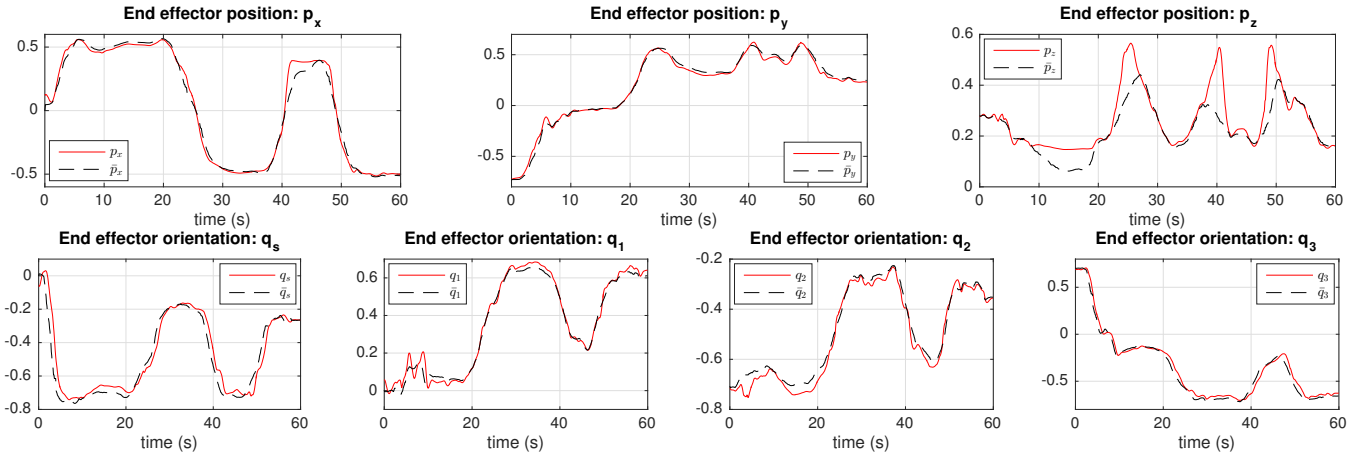

Fig. 5. Measured time evolution of: components p_x, p_y and p_z of the end effector position, and corresponding references $\bar{p}_x$, $\bar{p}_y$ and $\bar{p}_z$ (top row); components q_s, q_1, q_2 and q_3 of the quaternion representing the end effector orientation, and corresponding references $\bar{q}_s$, $\bar{q}_1$, $\bar{q}_2$ and $\bar{q}_3$ (bottom row).

The lower plot of Fig. 4 shows the values of δ_i (defined in (18)) for the three test points for Type-2 obstacles, which have to be greater or equal than 1. In this case, two of the test points (including the end effector) are on the threshold during certain time intervals. However, the value of δ_2 never violates the constraint. On the other hand δ_3 is slightly below the threshold for about 10 s in total, reaching a lower peak of 0.91. Also in this case, due to the presence of a safety zone around the obstacle, no actual collisions occur. Analogously to the upper plot, the lower plot displays the evolution of $\bar{\delta}_3$, i.e., the value that δ_3 would have if the reference were perfectly tracked: one can see that large violations of the constraint would occur. In the zoomed subfigure, we show the ability of MPC to exploit predictions to avoid large constraints violations: around $t = 39$ s, the rate of decrease of δ_3 diminishes around 1 s ($= T_p$) before the constraint is hit by the reference signal: this is due to the MPC controller slowing down the robot when a potential collision is foreseen. On the other hand, at time $t = 50$ s, δ_3 starts increasing 1 s before the corresponding reference value: this is due to the ability of the controller to exploit the prediction of the reference signal, which is correctly predicted as moving away from the obstacle.

Fig. 5 shows the tracking of all components of the end effector pose $\boldsymbol{x}_e$. In certain time intervals, due to the need to satisfy constraints, the control system allows an increase of the tracking error (this is mainly apparent on p_z), determining the best tracking performance achievable while satisfying the imposed constraints. The operator moves so that a perfect tracking would imply collision with the table between $t = 9$ s and $t = 20$ s: as a consequence, the tracking error for p_z is large in this time interval in order to satisfy the Type-1 obstacle constraint, as can be observed looking at the same time interval in the upper plot of Fig. 4. Notice that, specifically between $t = 15$ s and $t = 18$ s, the position reference remains approximately constant, and the MPC controller drives the robot end effector position as close as possible to the reference, while ensuring constraints satisfaction. After that, the operator provides an end-effector position reference that would imply collision with the computer case in a high percentage of the time interval between $t = 22$ s and $t = 60$ s (see the lower plot of Fig. 4). The price to pay for collision avoidance is visible mainly for variable p_z in Fig. 5, where the tracking error is large in certain time intervals. The tracking performance depends on the aggressiveness of the control law: a better tracking would have been obtained by decreasing the weights of the cost function for joint accelerations and velocities, but this would have increased the wear of the components and caused a jittery motion.

The human operator, in this case study, was aware of the presence of the MPC controller, and was therefore unaffected by the fact that the manipulator would move differently from the reference in the presence of active constraints. This can be noticed by the reference trajectories in Fig. 5, where it is apparent that no sudden movements of the operator hand happen in response to deviations of the robot from the nominal trajectory. In certain applications this lack of transparency can constitute a problem: a possibility to mitigate it would be the use of augmented reality [34], which would provide the operator with a visualization of the current reference that is being given to the robot end-effector.

One of the main aims of this work is to show that the presented experimental results can be obtained with standard computers available in any industrial setting, and can thus be applied to a wide range of real-world problems. For our case study, the average computation time for solving the MPC problem, consisting of 5 SQP steps, is 800 μs. More importantly, the maximum computation time, equal to 18.45 ms, is still well below the sampling interval length of 100 ms. The maximum value only occurs for one sampling instant, and, apart from that case, the computation time never exceeds 3 ms, as can be also seen in Fig. 6. The sudden increase in computation time happens in coincidence with the spikes in the value of joint

accelerations and velocities in Fig. 3. This is probably due to a change of the predicted hand pose reference, in the presence of active constraints. Indeed, at $t = 48$ s, the constraint on θ_4 is active (Fig. 3), together with that on the avoidance of Type-2 obstacles (bottom of Fig. 4). As can be seen in the bottom of Fig. 4, at $t = 48$ s the reference position of the end effector stops moving towards the inner part of the obstacle and starts moving towards outside. The simultaneous presence of these conditions probably caused the previous solution of the MPC problem not to be a good initialization for the current problem. As a consequence, a sudden change on velocities and accelerations occurred, and more calculations than usual were necessary for the solver to find the optimal control sequence.

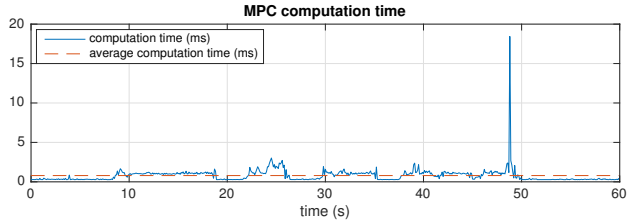

Fig. 6. Time evolution of the computation time for the MPC control law as a function of time.

V. Conclusions

This paper has described a general MPC framework for semi-autonomous teleoperation of robot manipulators with obstacle avoidance. The reported experimental results on the teleoperation of a UR5 robot confirmed the expected performance: the MPC controller allowed the manipulator to track the human hand pose while satisfying all the imposed constraints, and exploiting the prediction of the reference signals to proactively prevent collisions and improve the tracking performance. With a sampling interval of 100 ms and a prediction horizon of 1 s, the computation time never exceeded 19 ms, which makes the proposed method a possible candidate for solving semi-autonomous teleoperation problems in different scenarios. Future work will be devoted to extend the described approach to scenarios with dynamic obstacles, also developing MPC schemes with guaranteed stability.

References

[1] H. Boessenkool *et al.*, "A task-specific analysis of the benefit of haptic shared control during telemanipulation," *IEEE T Haptics*, vol. 6, no. 1, pp. 2–12, 2013.

[2] P. F. Hokayem and M. W. Spong, "Bilateral teleoperation: An historical survey," *Automatica*, vol. 42, no. 12, pp. 2035–2057, 2006.

[3] G. Salvietti *et al.*, "Multicontact bilateral telemanipulation with kinematic asymmetries," *IEEE/ASME T Mech*, vol. 22, no. 1, pp. 445–456, 2017.

[4] A. Flores-Abad *et al.*, "A review of space robotics technologies for on-orbit servicing," *Prog Aerosp Sci*, vol. 68, pp. 1–26, 2014.

[5] R. H. Taylor *et al.*, "Medical robotics and computer-integrated surgery," in *Springer handbook of robotics*, 2016, pp. 1657–1684.

[6] J. Vertut, *Teleoperation and robotics: applications and technology*. Springer Science & Business Media, 2013.

[7] C. Passenberg, A. Peer, and M. Buss, "A survey of environment, operator, and task-adapted controllers for teleoperation systems," *Mechatronics*, vol. 20, no. 7, pp. 787–801, 2010.

[8] D. Q. Mayne, "Model predictive control: Recent developments and future promise," *Automatica*, vol. 50, no. 12, pp. 2967–2986, 2014.

[9] B. Houska, H. J. Ferreau, and M. Diehl, "ACADO toolkit - an open-source framework for automatic control and dynamic optimization," *Optim Contr Appl Met*, vol. 32, no. 3, pp. 298–312, 2011.

[10] J. Mattingley and S. Boyd, "Cvxgen: A code generator for embedded convex optimization," *Optim Eng*, vol. 13, no. 1, pp. 1–27, 2012.

[11] L. Stella *et al.*, "A simple and efficient algorithm for nonlinear model predictive control," in *Proc IEEE Conf Dec Contr*, 2017, pp. 1939–1944.

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

[13] T. Faulwasser *et al.*, "Implementation of nonlinear model predictive path-following control for an industrial robot," *IEEE T Contr Syst T*, vol. 25, no. 4, pp. 1505–1511, 2017.

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

[15] A. Zhakatayev, M. Rubagotti, and H. A. Varol, "Closed-loop control of variable stiffness actuated robots via nonlinear model predictive control," *IEEE Access*, vol. 3, pp. 235–248, 2015.

[16] A. Jain *et al.*, "Reaching in clutter with whole-arm tactile sensing," *Int J Robot Res*, vol. 32, no. 4, pp. 458–482, 2013.

[17] M. D. Killpack, A. Kapusta, and C. C. Kemp, "Model predictive control for fast reaching in clutter," *Auton Robots*, vol. 40, no. 3, pp. 537–560, 2016.

[18] A. M. Jasour and M. Farrokhi, "Adaptive neuro-predictive control for redundant robot manipulators in presence of static and dynamic obstacles: A Lyapunov-based approach," *Int J Adaptive Contr Signal Proc*, vol. 28, no. 3-5, pp. 386–411, 2014.

[19] M. Wang, J. Luo, and U. Walter, "A non-linear model predictive controller with obstacle avoidance for a space robot," *Adv Space Res*, vol. 57, no. 8, pp. 1737–1746, 2016.

[20] A. Zube, "Cartesian nonlinear model predictive control of redundant manipulators considering obstacles," in *Proc IEEE Int Conf Ind Tech*, 2015, pp. 137–142.

[21] J. Koenemann *et al.*, "Whole-body model-predictive control applied to the HRP-2 humanoid," in *Proc IEEE/RSJ Int Conf Intell Rob Sys*, 2015, pp. 3346–3351.

[22] V. Kumar *et al.*, "Real-time behaviour synthesis for dynamic hand-manipulation," in *Proc IEEE Int Conf Robot Autom*, 2014, pp. 6808–6815.

[23] Z. Wang *et al.*, "Probabilistic movement modeling for intention inference in human-robot interaction," *Int J Robot Res*, vol. 32, no. 7, pp. 841–858, 2013.

[24] R. Chipalkatty, G. Droge, and M. B. Egerstedt, "Less is more: Mixed-initiative model-predictive control with human inputs," *IEEE T Robot*, vol. 29, no. 3, pp. 695–703, 2013.

[25] H. Liu and L. Wang, "Human motion prediction for human-robot collaboration," *J Manuf Syst*, vol. 44, pp. 287–293, 2017.

[26] B. Siciliano *et al.*, *Robotics: Modeling, Planning and Control*. Springer, 2010.

[27] P. Kumar and E. A. Yildirim, "Minimum-volume enclosing ellipsoids and core sets," *J Optimiz Theory App*, vol. 126, no. 1, pp. 1–21, 2005.

[28] D. Limon *et al.*, "Nonlinear MPC for tracking piece-wise constant reference signals," *IEEE T Autom Contr*, vol. 63, no. 11, pp. 3735–3750, 2018.

[29] T. Taunyazov, B. Omarali, and A. Shintemirov, "A novel low-cost 4-DOF wireless human arm motion tracker," in *Proc IEEE Int Conf Biom Robot Biomech*, 2016, pp. 157–162.

[30] T. T. Andersen, "Optimizing the Universal Robots ROS driver." Technical University of Denmark, Tech. Rep., 2015.

[31] Y. Liu and Y. Zhang, "Towards welding robot with human knowledge: A remotely-controlled approach," *IEEE T Autom Sci Eng*, vol. 12, no. 2, pp. 769–775, 2015.

[32] K. Mathiassen *et al.*, "An ultrasound robotic system using the commercial robot UR5," *Frontiers in Robotics and AI*, vol. 3, 2016.

[33] H. J. Ferreau *et al.*, "qpOASES: A parametric active-set algorithm for quadratic programming," *Math Progr Comp*, vol. 6, no. 4, pp. 327–363, 2014.

[34] M. Billinghurst *et al.*, "A survey of augmented reality," *Found Trends Human–Computer Interact*, vol. 8, no. 2-3, pp. 73–272, 2015.