# Accurate sensorless lead-through programming for lightweight robots in structured environments

Matteo Ragaglia, Andrea Maria Zanchettin, Luca Bascetta, Paolo Rocco

*Politecnico di Milano, Dipartimento di Elettronica, Informazione e Bioingegneria, Piazza Leonardo Da Vinci 32, Milano, Italy*

ABSTRACT

Nowadays, programming an industrial manipulator is a complex and time-consuming activity, and this prevents industrial robots from being massively used in companies characterized by high production flexibility and rapidly changing products. The introduction of sensor-based lead-through programming approaches (where the operator manually guides the robot to teach new positions), instead, allows to increase the speed and reduce the complexity of the programming phase, yielding an effective solution to enhance flexibility. Nevertheless, some drawbacks arise, like for instance lack of accuracy, need to ensure the human operator safety, and need for force/torque sensors (the standard devices adopted for lead-through programming) that are expensive, fragile and difficult to integrate in the robot controller.

This paper presents a novel approach to lead-through robot programming. The proposed strategy does not rely on dedicated hardware since torques due to operator's forces are estimated using a model-based observer fed with joint position, joint velocity and motor current measures. On the basis of this information, the external forces applied to the manipulator are reconstructed. A voting system identifies the largest Cartesian component of the force/torque applied to the manipulator in order to obtain accurate lead-through programming via admittance control. Finally an optimization stage is introduced in order to track the joint position displacements computed by the admittance filter as much as possible, while enforcing obstacle avoidance constraints, actuation bounds and Tool Centre Point (TCP) operational space velocity limits. The proposed approach has been implemented and experimentally tested on an ABB dual-arm concept robot FRIDA.

*Keywords:*
Lead-through Sensorless
Robot programming Safety
Robot teaching

## 1. Introduction

Nowadays industrial robots are able to offer fast and accurate task execution in various industrial fields, but they are inherently characterized by a low degree of adaptability to rapidly changing task specifications. The fact that programming an industrial manipulator is a complex and time-consuming activity represents one of the main weaknesses of today's industrial robotic systems, since it is preventing industrial manipulators from being massively used in SMEs, where small size production and rapidly changing product features request the highest production flexibility.

The introduction of lead-through programming (from now on "LTP") approaches [1,2] has definitely helped increasing the speed and reducing the complexity of the programming phase, by allowing the human operator to manually guide the robot in order to teach new positions. Nevertheless, these solutions present some drawbacks as well. Not only lead-through programming cannot guarantee the same level of accuracy achievable with a teach pendant-based programming, but it can be also potentially unsafe since it requires physical Human–Robot Interaction (pHRI).

Moreover, it is difficult to customize LTP in terms of both selecting directions of motion and/or enforcing speed limits. Although the possibility of constraining the movement of the robot during LTP has been addressed in the field of surgical robotics [3–5], the proposed approaches consist in simply defining a safety envelope from which the manipulator end-effector cannot exit, while it is guided by the human. Only at a later stage [6], the formalization of more generic constraints for lead-through programming of surgical robots was introduced.

Furthermore LTP relies on dedicated hardware, i.e. force/torques sensors. Although various manipulators have been designed by introducing force/torques sensors and/or compliant joints [7–11], the great majority of industrial manipulators is not inherently equipped with hardware that enables LTP. Adding a force/torque sensor to a standard industrial robot is a rather expensive and difficult operation.

*E-mail addresses:* matteo.ragaglia@polimi.it (M. Ragaglia), andreamaria.zanchettin@polimi.it (A.M. Zanchettin), luca.bascetta@polimi.it (L. Bascetta), paolo.rocco@polimi.it (P. Rocco).

In order to overcome this particular limitation, several approaches to the problem of sensorless detection of human–robot physical contact have been proposed in the literature, mainly based on fault detection and isolation algorithms fed with motor torque (or alternatively motor current) measurements [12]. As an example, the ABB RobotWare Software running on the IRC5 industrial controller includes this kind of feature [13], while Geravand et al. [14] present a safe collision detection and reaction strategy developed on an industrial manipulator with a closed control architecture.

Nevertheless, sensorless collision detection is not sufficient to completely overcome the need for dedicated hardware, since real-time estimation of the values achieved by interaction forces applied by the human operator to the manipulator (and not just their occurrence) is obviously necessary to perform sensorless LTP.

The problem of sensorless real-time estimation of external forces (or torques due to these forces) has been addressed in the literature, as well. The most relevant proposed solutions (like for instance [15–20]) rely on the calculation of residuals based on the manipulator generalized momentum. Typically estimation of external torques is used as input for impedance and/or admittance control strategies, like for instance in [21,22]. Other strategies for contact force estimation based on both friction estimation and detuning of the low-level joint control loops, have been proposed in [23–25].

While several force/torque estimation strategies have been developed in order to avoid the use of dedicated sensors, the possibility to apply these strategies in the field of LTP has not been thoroughly explored yet. As a matter of fact, a complete and well-established framework for sensorless, accurate and safe LTP of a typical industrial manipulator in a structured environment is still missing. This possibility to teach an industrial robot by manually driving it without the need of dedicated hardware and taking into account motion accuracy and, above all, operator's safety, definitely represents an interesting enhancement in both the fields of robot programming and pHRI.

In this sense, the main contribution of this work consists in introducing a novel LTP strategy for industrial manipulators. This strategy combines the positive aspects of traditional teach pendant-based techniques (accuracy, high level of customization, safety and no need for additional hardware) with the advantages brought by lead-through techniques (reduced programming time and user friendliness), while mitigating their respective limitations.

More in depth, the key innovative features of the proposed LTP strategy can be summarized as follows:

- *Accuracy*: A voting system and a Finite State Machine (FSM) work together to select the largest Cartesian component of the forces/moments applied by the operator, thus allowing him/her to modify only one operational space degree of freedom at a time;
- *Safety*: The output of the admittance filter is processed by an optimization stage in order to satisfy actuation bounds, safety-related limits on operational space TCP velocity and avoidance of known obstacles. Moreover, a redundancy resolution algorithm ensures that the entire manipulator kinematic chain does not collide with workspace objects.

Finally, from a methodological point of view, the formalization of the “safety constraints”, originally presented in [26], is here extended from the case of point-shaped obstacles to the case of arbitrarily-shaped convex obstacles.

The remainder of this work is organized as follows. The formulation of the manipulator dynamic model is described in Section 2. Section 3 presents a detailed explanation of the proposed programming algorithm, whose main building blocks are the model-based estimator of external interaction forces/moments, the voting system, the FSM, the admittance filter, the optimization stage, and the redundancy resolution criterion. Section 4 introduces the experimental setup and presents the validation of the proposed programming strategy performed on the ABB dual-arm concept robot FRIDA (see Fig. 1). Finally, Section 5 discusses the obtained results and presents possible developments.

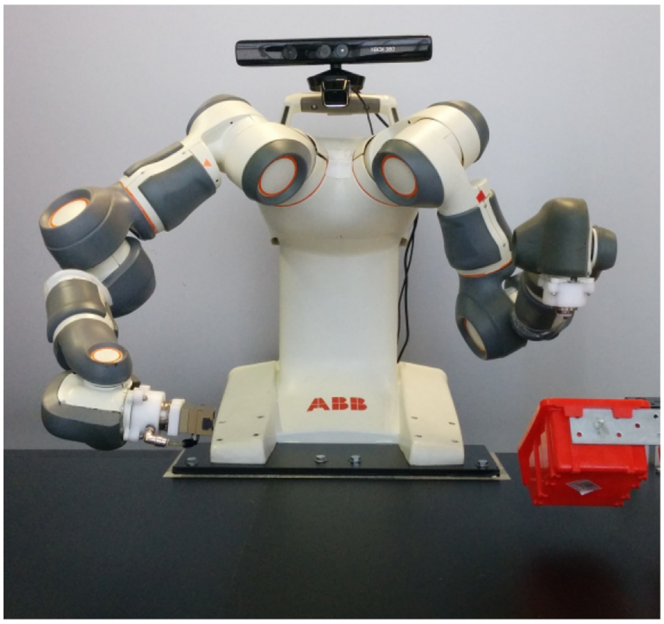


**Fig. 1.** ABB prototype robot FRIDA. The picture also shows the experimental setup for Experiment #1 and Experiment #2.

## 2. Manipulator dynamic model

The manipulator dynamic model is expressed, in Euler–Lagrange formulation, by the following equation:

$$\boldsymbol{B}(\boldsymbol{q})\ddot{\boldsymbol{q}} + \boldsymbol{C}(\boldsymbol{q},\ \dot{\boldsymbol{q}})\dot{\boldsymbol{q}} + \boldsymbol{G}(\boldsymbol{q}) + \boldsymbol{F}(\dot{\boldsymbol{q}}) = \boldsymbol{\tau} + \boldsymbol{\tau}_{ext} \tag{1}$$

where $\boldsymbol{q}$, $\dot{\boldsymbol{q}}$ and $\ddot{\boldsymbol{q}}$ represent joint positions, velocities and accelerations, respectively, $\boldsymbol{g}$ is the gravity acceleration vector, $\boldsymbol{B}(\boldsymbol{q})$, $\boldsymbol{C}(\boldsymbol{q},\dot{\boldsymbol{q}})$ and $\boldsymbol{G}(\boldsymbol{q})$ represent the inertia matrix, the Coriolis-centrifugal matrix and the gravitational vector, term respectively, $\boldsymbol{F}(\dot{\boldsymbol{q}})$ is the function modelling friction torques, $\boldsymbol{\tau}$ denotes motor torques. $\boldsymbol{\tau}_{ext}$ represents the external interaction torques due to forces/moments $\boldsymbol{\mu}$ applied to the manipulator, given by the following relation:

$$\boldsymbol{\tau}_{ext} = \boldsymbol{J}_C(\boldsymbol{q})^T \boldsymbol{\mu} \tag{2}$$

where $\boldsymbol{J}_C(\mathbf{q})$ is the Jacobian associated to the contact point $C$.

DH kinematic parameters and masses of the links, gear ratios, centre of gravity positions, inertia tensors, and motor inertias are assumed to be a priori known from manufacturer data-sheets. A very simple, yet effective, friction model has been chosen. The model consists of a viscous friction term plus a static friction component. A linear approximation of the discontinuous static friction is adopted for joint velocities in a range $|\dot{q}_i| \le 0.01$. Identification of friction coefficients is performed via Least Square error minimization with non-negativity constraints. Section 4.1 provides validation results that confirm the accuracy of both the known and the identified parameters for the given case.

Although it is outside the scope of this work, it would be possible to consider more sophisticated friction models in order to improve the accuracy of our dynamic model, like for instance the friction observer proposed in [27]. More recently, in [28], the authors approach the problem of external force estimation when the robot is still. Since in this scenario static friction significantly affects the estimation of applied forces, a dithering feed-forward torque is used to improve the estimation accuracy.

## 3. Lead-through programming algorithm

As mentioned before, the proposed LTP strategy represents a significant improvement with respect to state-of-the-art robot programming techniques not only because it overcomes the need for additional hardware, but also as it combines user-friendliness with motion accuracy, safety features and with the possibility of customizing robot movements by imposing different kinds of constraints. In the following a high level description of the proposed programming method is reported and each fundamental building block is thoroughly detailed.

First of all, considering the block scheme depicted in Fig. 2, in order to perform sensorless LTP, applied external forces and moments need to be estimated on the basis of all the information that can be directly acquired from the robot: joint positions, joint velocities and motor torques (see Section 3.1). Then, to achieve accuracy of the enforced motion, it is necessary to identify the largest Cartesian component of the estimated forces/moments in order to select the proper direction. To achieve this goal a voting system is introduced together with a Finite State Machine that outputs a projection of the estimated forces/moments according to the output of the voting system itself (see Section 3.2).

At this point, projected external forces/moments are transformed into operational space reference positions and velocities by using an admittance filtering techniques (see Section 3.3).

Finally, the constraint-based controller computes the new reference values for joint positions and velocities (Section 3.5) by minimizing the error with respect to the output of the admittance filters while obeying to different types of constraints: actuation bounds, safety-inspired limits on operational space TCP velocity, and avoidance of known obstacles (whose derivation is detailed in Section 3.4).

### *3.1. Online external forces/moments estimation*

In order to estimate the external forces/moments $\boldsymbol\mu$ applied to the manipulator it is necessary to compute (or to estimate) in real-time the external torques $\boldsymbol\tau_{ext}$. Unfortunately, while the Euler–Lagrange formulation of the dynamic model given in Eq. (1) can be successfully exploited to perform offline identification of unknown dynamic parameters, it is not possible to rely on this formulation for real-time estimation of interaction forces/moments, since accurate online computation of joint accelerations by numerical differentiation of joint velocities (or either from double numerical differentiation of joint positions) is not feasible.

Considering the generalized moments $\boldsymbol p = \boldsymbol B(\boldsymbol q)\,\dot{\boldsymbol q}$ and their derivative:

$$\dot{\boldsymbol p} = \boldsymbol C(\boldsymbol q, \dot{\boldsymbol q})^T \dot{\boldsymbol q} - \boldsymbol G(\boldsymbol q) - \boldsymbol F(\dot{\boldsymbol q}) + \boldsymbol\tau + \boldsymbol\tau_{ext} \tag{3}$$

it is possible to define the residuals [15] at time $t_k$ as

$$\boldsymbol r(t_k) = \boldsymbol K\left[\boldsymbol p(t) - \boldsymbol p(0) - \int_0^{t_k}\left(\boldsymbol\tau + \boldsymbol C(\boldsymbol q, \dot{\boldsymbol q})^T\dot{\boldsymbol q} - \boldsymbol G(\boldsymbol q) - \boldsymbol F(\dot{\boldsymbol q}) + \boldsymbol r(t)\right)dt\right] \tag{4}$$

where $\boldsymbol K$ is a diagonal positive definite matrix. By computing (4), one obtains a first order stable linear relationship between the external torques $\boldsymbol\tau_{ext}$ and the defined residual vector

$$\dot{\boldsymbol r} = \boldsymbol K(\boldsymbol\tau_{ext} - \boldsymbol r) \tag{5}$$

Assuming all interaction forces/moments $\boldsymbol\mu$ are applied to the TCP frame, the following equation:

$$\boldsymbol\mu = \left(\boldsymbol J(\boldsymbol q)^T\right)^{\dagger}\boldsymbol r \tag{6}$$

can be finally adopted to estimate the applied force/momentum $\boldsymbol\mu$, where $\boldsymbol J$ represents the Jacobian of the TCP frame, while the symbol $\boldsymbol A^{\dagger}$ stands for the Moore–Penrose pseudo-inverse of matrix $\boldsymbol A$. By exploiting the knowledge of the dynamic model and the measurement of the applied motor torques $\boldsymbol\tau$, it is possible to achieve an estimate, at time $t_k$, of the applied interaction forces/moments $\boldsymbol\mu$ through (5) and (6).

For real-time implementation, the observer of the forces/moments is discretized in the following way:

$$\begin{aligned}
&\boldsymbol v_0 = \boldsymbol 0\\
&\boldsymbol p_k = \boldsymbol B(\boldsymbol q_k)\,\dot{\boldsymbol q}_k\\
&\boldsymbol r_k = (\boldsymbol I + \Delta t\boldsymbol K)^{-1}\boldsymbol K\left[\boldsymbol p_k - \boldsymbol p_0 - \boldsymbol v_k - \dot{\boldsymbol p}_k\,\Delta t\right]\\
&\boldsymbol v_{k+1} = \boldsymbol v_k + \left(\dot{\boldsymbol p}_k + \boldsymbol r_k\right)\Delta t\\
&\boldsymbol\mu_k = \left(\boldsymbol J(\boldsymbol q_k)^T\right)^{\dagger}\boldsymbol r_k
\end{aligned} \tag{7}$$

where $\boldsymbol v_k$ is the state variable of the observer at time $t_k$ and $\Delta t$ is the discrete time step.

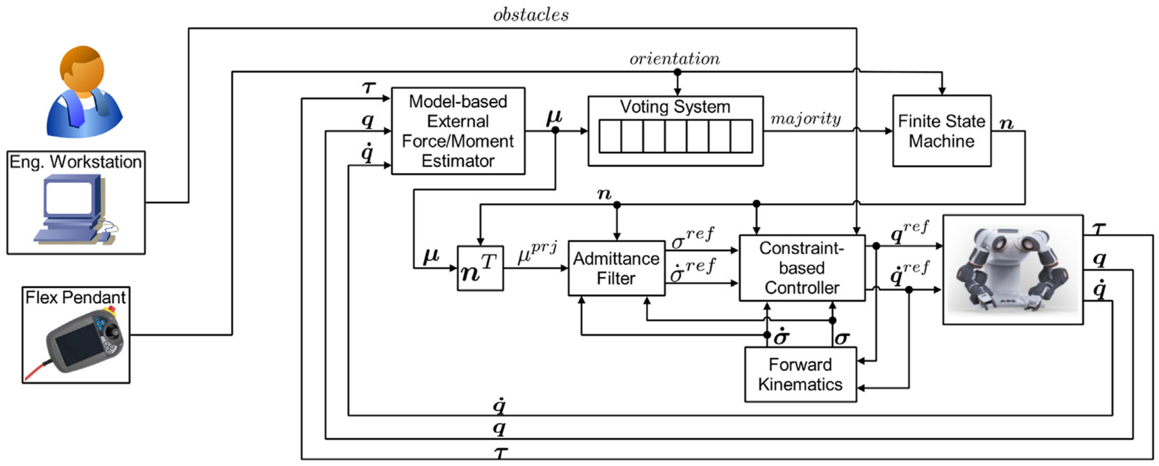


**Fig. 2.** Block diagram of the proposed algorithm for sensorless, accurate and safe lead-through programming.

## 3.2. Voting system and finite state machine

The voting system and the Finite State Machine (FSM) represented in Fig. 2 play a crucial role in the context of the proposed LTP strategy, since they allow to enforce accurate end-effector motion by selecting both the largest Cartesian component of the applied force/momentum and the corresponding Cartesian direction.

We assume that the human operator wants to move/rotate the end-effector along/around a certain Cartesian direction, while applying a force/momentum approximately directed along/around such direction. Therefore we identify the largest Cartesian component of the applied force/momentum and we force the manipulator end-effector to move/rotate exclusively along/around the corresponding Cartesian axis.

More in depth, the voting system consists in a circular buffer that collects votes. These votes are assigned to the three Cartesian axis *X*, *Y* and *Z* on the basis of two distinct inputs: the estimated applied forces/moments $\boldsymbol{\mu}$ and a boolean flag, named "*orientation*", that can be directly set by the human operator in order to enforce either translational or rotational motion. At each iteration, if $\boldsymbol{\mu}$ is equal to the null vector, a blank vote is cast and inserted into the buffer. Otherwise, if *orientation* is equal to 0, a vote is assigned to the axis corresponding to the largest component of the estimated force, while if *orientation* is set to 1, the vote is cast for the direction corresponding to the largest component of the applied momentum.

When a significantly large number of votes is assigned to the same Cartesian axis, we can assume that the operator wants to enforce a translational/rotational motion along/around that direction. Consequently, the voting system outputs a discrete variable, named "*majority*", that can assume four different values: "*null*", "*X*", "*Y*", "*Z*". Obviously each value corresponds to a Cartesian axis, a part from *null* that means that either no force/momentum is exerted on the robot or that no Cartesian direction is assigned a sufficiently large number of votes.

More in detail, whenever *majority* is equal to *null* and the number of votes assigned to a Cartesian direction reaches a threshold value of $maj_{set}$ votes, the value of *majority* is set to the corresponding axis. On the other hand, if *majority* is not equal to *null* and the number of votes assigned to the corresponding Cartesian axis decreases below the threshold value of $maj_{reset}$ votes, *majority* is reset to *null*. Both $maj_{set}$ and $maj_{reset}$ are parametric thresholds that can be set by the human operator. Finally, whenever the operator modifies the value of the *orientation* flag, the buffer is emptied and *majority* is set to *null*.

Both *orientation* and *majority* act as input to the FSM, whose state-transition graph is represented in Fig. 3. On the basis of its internal state, the FSM computes a specific value of the unit projection vector $\boldsymbol{n}$ (as reported in Fig. 3), thus determining which component of the estimated applied force/momentum will be passed to the next stages of the programming algorithm.

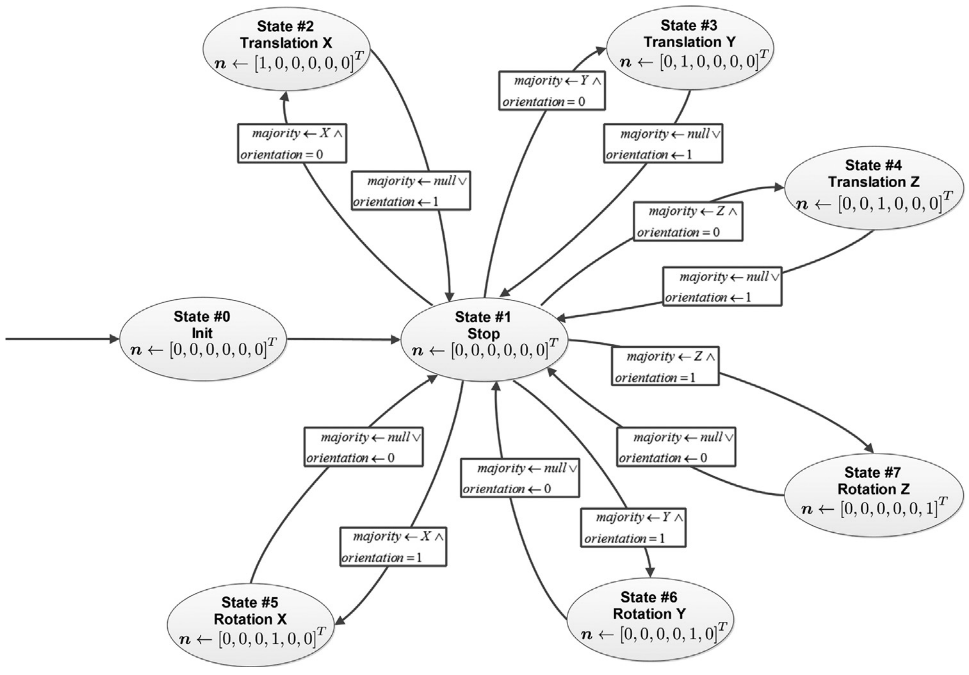
Then the estimated applied forces/moments $\boldsymbol{\mu}$ are projected onto $\boldsymbol{n}$ to obtain a scalar quantity $\mu^{prj}$:

$$\mu^{prj} = \boldsymbol{n}^T \boldsymbol{\mu} \tag{8}$$

Clearly, both in State #0 and in State #1 the resulting projection vector zeroes all the estimated forces/moments, either stopping the manipulator or keeping it still until a new value of *majority* is set by the voting system. Otherwise, by sending both $\mu^{prj}$ and $\boldsymbol{n}$ to the admittance filtering stage of the algorithm, it is possible to obtain an accurate translational/rotational motion of the end-effector along/around only one Cartesian axis at a time.

Necessarily, whenever the FSM is in State #1, and the operator wants to move the end-effector along a specific direction, the voting system has to wait for a new value of *majority* to be set before the FSM can switch to the desired State, thus introducing some delay between the human action and the robot reaction. This delay depends on the length of the buffer included in the voting system. During validation experiments we tested different buffer lengths: 100, 75 and 50 votes. Given these dimensions, the minimum time needed to establish an absolute majority among the votes is, respectively, 0.20 s, 0.15 s and 0.10 s.

It is worth noting that this delay is not sufficiently large to influence the human operator's behaviour (see the attached video for a practical example). Moreover, the maximum delay only occurs when the operator wants to change the direction of motion and a new majority has to be established.



State #2 Translation X $\boldsymbol{n} \leftarrow [1,0,0,0,0,0]^T$
State #3 Translation Y $\boldsymbol{n} \leftarrow [0,1,0,0,0,0]^T$
State #4 Translation Z $\boldsymbol{n} \leftarrow [0,0,1,0,0,0]^T$
State #0 Init $\boldsymbol{n} \leftarrow [0,0,0,0,0,0]^T$
State #1 Stop $\boldsymbol{n} \leftarrow [0,0,0,0,0,0]^T$
State #7 Rotation Z $\boldsymbol{n} \leftarrow [0,0,0,0,0,1]^T$
State #5 Rotation X $\boldsymbol{n} \leftarrow [0,0,0,1,0,0]^T$
State #6 Rotation Y $\boldsymbol{n} \leftarrow [0,0,0,0,1,0]^T$
$majority \leftarrow X \wedge orientation = 0$
$majority \leftarrow null \vee orientation \leftarrow 1$
$majority \leftarrow Y \wedge orientation = 0$
$majority \leftarrow null \vee orientation \leftarrow 1$
$majority \leftarrow Z \wedge orientation = 0$
$majority \leftarrow null \vee orientation \leftarrow 1$
$majority \leftarrow Z \wedge orientation = 1$
$majority \leftarrow null \vee orientation \leftarrow 0$
$majority \leftarrow null \vee orientation \leftarrow 0$
$majority \leftarrow null \vee orientation \leftarrow 0$
$majority \leftarrow X \wedge orientation = 1$
$majority \leftarrow Y \wedge orientation = 1$

**Fig. 3.** FSM state transition diagram.

### 3.3. Admittance filtering

The previously computed projection of the estimated applied forces/moments $\mu^{prj}$ is processed by an admittance filtering stage in order to convert forces into linear velocities and moments into angular velocities. These, in turn, modify the end-effector operational space configuration $\boldsymbol{\sigma}$, defined as the output of the forward kinematics of the manipulator:

$$\boldsymbol{\sigma} = \boldsymbol{\sigma}(\boldsymbol{q}) = \left[x,\ y,\ z,\ \varphi,\ \theta,\ \psi\right]^T \tag{9}$$

where $x$, $y$ and $z$ represent the end-effector position coordinates in the Cartesian space and $\mathbf{\Phi} = [\phi, \theta, \psi]^T$ is a set of Euler angles describing the end-effector orientation.

Following the formulation of the operational space admittance given in [29], two distinct admittance filter are considered: a positional admittance filter and a rotational one.

#### 3.3.1. Positional admittance filtering

In case the *orientation* flag is equal to 0, the projection of the estimated applied forces/torques $\mu^{prj}$ corresponds to the principal Cartesian component of the applied external force and the positional admittance filter converts $\mu^{prj}$ in an operational space linear velocity reference $\dot{\sigma}^{ref}$, determining an operational space translational displacement reference $\sigma^{ref}$ along the Cartesian direction selected by the voting system and the FSM.

The admittance dynamic relation is defined as follows:

$$\mu^{prj} = M_p \ddot{\sigma}^{ref} + D_p \dot{\sigma}^{ref} \tag{10}$$

Obviously, the relation is stable, provided that constants $M_p$ and $D_p$ are positive.

Considering a second-order Taylor discretization, it is possible to rewrite the admittance relation as follows:

$$\sigma_{k+1}^{ref} = \sigma_k^{ref} + \left(\Delta t - \frac{\Delta t^2}{2}\frac{D_p}{M_p}\right)\dot{\sigma}_k^{ref} + \frac{\Delta t^2}{2}\frac{1}{M_p}\mu_k^{prj} \tag{11}$$

$$\dot{\sigma}_{k+1}^{ref} = \left(1 - \Delta t\,\frac{D_p}{M_p}\right)\dot{\sigma}_k^{ref} + \Delta t\,\frac{1}{M_p}\mu_k^{prj} \tag{12}$$

where $\mu_k^{prj}$ represents the projection of the estimated applied forces/torques computed at time $t_k$ and $\Delta t$ is the discrete time step.

Finally, the switching between different translational motions can be easily managed by the positional admittance filter, thanks to the FMS. As a matter of fact, in order to change the direction of the enforced motion, first the FSM internal state must be reset to State #1 and then it must be set to State #2, State #3, or State #4 (see again Fig. 3) on the basis of the applied forces. Since the transition to State #1 requires the robot to stop, the new translational motion necessarily starts with the manipulator being still, thus allowing the admittance filter to handle the switching by simply setting both $\dot{\sigma}^{ref}$ and $\ddot{\sigma}^{ref}$ equal to zero, before starting to compute the new velocity and position references along the Cartesian direction selected by the new value of the projection vector $\boldsymbol{n}$.

#### 3.3.2. Rotational admittance filtering

When the *orientation* flag is set to 1, the projection of the estimated applied forces/torques $\mu^{prj}$ consists in the largest Cartesian component of the applied external momentum. In this case, the rotational admittance filter converts $\mu^{prj}$ in an operational space angular velocity reference $\dot{\sigma}^{ref}$, determining an operational space angular displacement reference $\sigma^{ref}$ for a specific Euler angle.

The rotational admittance dynamic relation can be defined in the following way:

$$\mu^{prj} = M_\Phi \ddot{\sigma}^{ref} + D_\Phi \dot{\sigma}^{ref} \tag{13}$$

Once again, the relation is stable when constants $M_\Phi$ and $D_\Phi$ are positive, but, differently from the positional case, there is no bijective relation between the Cartesian component of the applied torque and the Euler angles first-order derivatives $\dot{\mathbf{\Phi}}$. Consequently, it is not possible to use a unique representation of the end-effector frame orientation in order to convert an external torque applied on a specific Cartesian direction to an angular velocity of the frame around the same axis. Instead, three different Euler angles representation are chosen.

When the output of the FSM is $\boldsymbol{n} = [0,\ 0,\ 0,\ 1,\ 0,\ 0]^T$, $\mu^{prj}$ corresponds to the $X$-axis component of the applied momentum. By defining the end-effector frame angular velocity $\boldsymbol{\omega}$ in the following way:

$$\boldsymbol{\omega} = [\omega_X, \omega_Y, \omega_Z]^T \tag{14}$$

the end-effector frame rotation around the reference frame $X$-axis that must be enforced is described by

$$\boldsymbol{\omega} = \left[\omega_X, 0, 0\right]^T \tag{15}$$

Considering $XZX$ Euler angles $\mathbf{\Phi}_{XZX} = [\phi_X, \theta_Z, \psi_X]^T$, it is possible to express the angular velocity $\boldsymbol{\omega}$ as a function of the Euler angles first-order derivatives $\dot{\mathbf{\Phi}}_{XZX} = [\dot{\phi}_X, \dot{\theta}_Z, \dot{\psi}_X]^T$:

$$\boldsymbol{\omega} = \boldsymbol{W}(\Phi_{XZX})\dot{\mathbf{\Phi}}_{XZX} \tag{16}$$

$$\boldsymbol{W}(\Phi_{XZX}) = \begin{bmatrix} 1 & 0 & \cos\theta_Z \\ 0 & -\sin\varphi_X & \cos\varphi_X \sin\theta_Z \\ 0 & \cos\varphi_X & \sin\varphi_X \sin\theta_Z \end{bmatrix} \tag{17}$$

Consequently, by imposing $\dot{\theta}_Z = \dot{\psi}_X = 0$, $\omega_X$ is equal to:

$$\omega_X = \left[1,\ 0,\ \cos\theta_Z\right]\begin{bmatrix}\dot{\varphi}_X \\ \dot{\theta}_Z \\ \dot{\psi}_X\end{bmatrix} = \dot{\varphi}_X \Longrightarrow \dot{\omega}_X = \ddot{\varphi}_X \tag{18}$$

As a consequence a one-to-one correspondence is established between the end-effector angular velocity around the $X$-axis and the first-order derivative of a specific Euler angle, allowing us to set $\dot{\sigma}^{ref} = \dot{\phi}_X$ and $\ddot{\sigma}^{ref} = \ddot{\phi}_X$ inside the rotational admittance relation (13) in order to generate the desired rotational motion.

Similarly, in case the motion to be enforced is a rotation of the end-effector frame around the $Y$-axis ($\boldsymbol{n} = [0,\ 0,\ 0,\ 0,\ 1,\ 0]^T$), $YXY$ Euler angles are chosen, while when the desired motion consists in a rotation of the end-effector frame around the $Z$-axis ($\boldsymbol{n} = [0,\ 0,\ 0,\ 0,\ 0,\ 1]^T$) the $ZYZ$ set of Euler angles is employed.

Once again, it is possible to discretize the rotational admittance relation (13) by simply substituting $M_\Phi$ and $D_\Phi$ for $M_p$ and $D_p$, respectively, into Eqs. (11) and (12). Finally, the rotational admittance filter manages the switching between different rotational motion by stopping the robot before a new rotation axis is selected, depending on the applied moments.

### 3.4. Avoidance of known obstacles

In structured environments (i.e. in the presence of obstacles located inside the robot workspace) a problem that can easily arise while performing LTP is represented by collisions. In order to ensure that the entire kinematic chain of the manipulator performs a collision-free motion during LTP, safety constraints can be defined taking into account the kinematic configuration of the manipulator, known obstacles positions and their geometry.

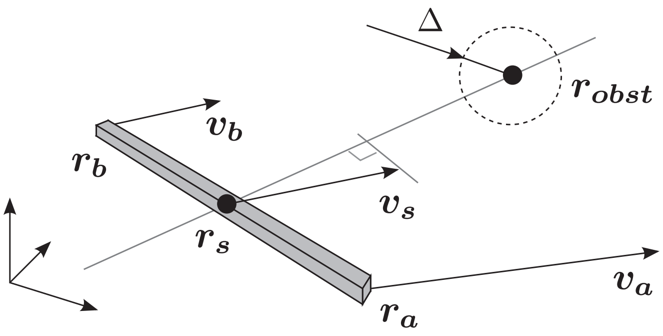


**Fig. 4.** A rigid beam representing one link.

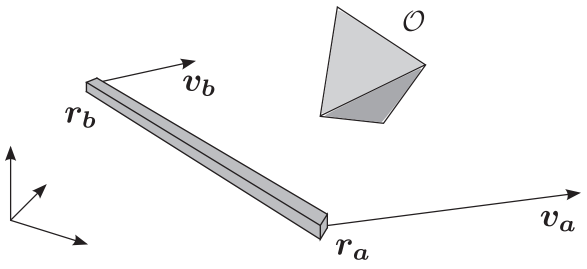


**Fig. 5.** A generic polytopic (convex) obstacle.

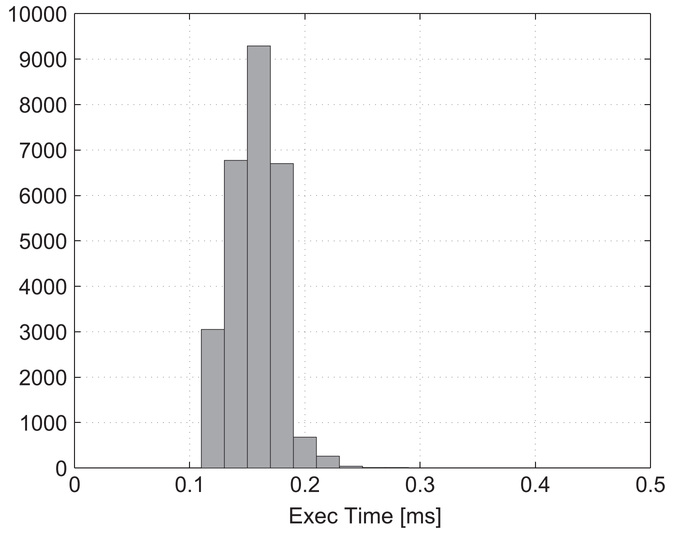


**Fig. 6.** Histogram showing the measured execution times of more than 25,000 calls to qpOASES.

**Table 1**
Optimization stage execution time: average execution time, standard deviation, minimum and maximum time.

| Exec. time | |
|---|---|
| Mean [ms] | 0.1569 |
| Std. [ms] | 0.0215 |
| Min. [ms] | 0.1140 |
| Max. [ms] | 0.2840 |

Consider Fig. 4 which represents a point obstacle $\boldsymbol{r}_{obst}$ as well as a generic robotic link, whose endpoints are at positions $\boldsymbol{r}_a$ and $\boldsymbol{r}_b$.

At all time, the robot trajectory must obey the following safety requirement expressed as an inequality among the robot velocity, its distance from an obstacle and a clearance parameter:

**Table 2**
Results of FRIDA's Dynamic Model Validation. For each joint: average error, maximum absolute error and standard deviation of the error with respect to maximum joint torques.

| Joint | Avg. error [%] | Max error [%] | Std. deviation [%] |
|---|---|---|---|
| 1 | −5.2867 | 12.00 | 4.90 |
| 2 | −7.7000 | 10.00 | 3.31 |
| 3 | +6.2283 | 15.00 | 6.35 |
| 4 | −3.1350 | 12.00 | 4.73 |
| 5 | −2.5000 | 15.00 | 7.55 |
| 6 | +1.1867 | 15.00 | 7.05 |
| 7 | −3.7667 | 15.00 | 6.43 |

$$\text{velocity}\cdot T_s \leq \max(0, \text{distance} - \text{clearance})$$

where the braking time $T_s$ possibly depends on the robot payload [30]. For a generic point $\boldsymbol{r_s}$ on the robot link, with velocity $\boldsymbol{v_s}$, we have:

$$\boldsymbol{v}_s^T \frac{\boldsymbol{r}_{obst} - \boldsymbol{r}_s}{\|\boldsymbol{r}_{obst} - \boldsymbol{r}_s\|} T_s \leq \max(0, \|\boldsymbol{r}_{obst} - \boldsymbol{r}_s\| - \Delta) \tag{19}$$

where $\Delta$ is a clearance parameter. According to Zanchettin and Rocco [26], if we assume the following parameterization of the link in terms of position and velocity of its end points

$$\boldsymbol{r}_s = \boldsymbol{r}_a + s(\boldsymbol{r}_b - \boldsymbol{r}_a), \quad \boldsymbol{v}_s = \boldsymbol{r}_b + s(\boldsymbol{v}_s - \boldsymbol{v}_a) \tag{20}$$

it is possible to manipulate inequality (19) and to write the minimum separation distance criterion in matrix form as

$$T_s \boldsymbol{E}\dot{\boldsymbol{q}} \leq \boldsymbol{f} \tag{21}$$

where

$$\boldsymbol{E} = \begin{bmatrix} (\boldsymbol{r}_{obst} - \boldsymbol{r}_a)^T \boldsymbol{J}_a \\ (\boldsymbol{r}_{obst} - \boldsymbol{r}_a)^T \boldsymbol{J}_b - (\boldsymbol{r}_b - \boldsymbol{r}_a)^T \boldsymbol{J}_a \end{bmatrix}$$

$$\boldsymbol{f} = \left[ \max\left(0, \min_s \|\boldsymbol{r}_{obst} - \boldsymbol{r}_s\| - \Delta\right)\right]^2 \begin{bmatrix} 1 \\ 1 \end{bmatrix} \tag{22}$$

$\boldsymbol{J}_a$ and $\boldsymbol{J}_b$ are position Jacobians of the two link end points.

The representation of safety constraints can be extended to the case of obstacles having more complex geometry. Consider a generic polytopic obstacle $\mathcal{O}$ as shown in Fig. 5. The constraints to be enforced for such an obstacle can be written as follows:

$$T_s \boldsymbol{E}(\boldsymbol{r}_{obst})\dot{\boldsymbol{q}} \leq \boldsymbol{f}(\boldsymbol{r}_{obst}), \quad \forall\, \boldsymbol{r}_{obst} \in \mathcal{O} \tag{23}$$

The number of constraints to be enforced at run time is conceptually infinite, i.e. one per each point belonging to $\mathcal{O}$. However, some geometrical properties of the obstacle can be exploited in order to make the problem tractable.

A sufficient condition for (23) to be satisfied for all points $\boldsymbol{r}_{obst} \in \mathcal{O}$ is

$$T_s \boldsymbol{E}(\boldsymbol{r}_{obst})\dot{\boldsymbol{q}} \leq d \begin{bmatrix} 1 \\ 1 \end{bmatrix}, \quad \forall\, \boldsymbol{r}_{obst} \in \mathcal{O} \tag{24}$$

where the right hand side term $d = \min_{\boldsymbol{r}_{obst} \in \mathcal{O}} \|\boldsymbol{f}(\boldsymbol{r}_{obst})\|_\infty$ represents the minimum distance between the link of the robot and the polytopic obstacle $\mathcal{O}$ and can be easily computed using the GJK algorithm [31]. Moreover, notice that the left hand side term is linear with respect to the parameter $\boldsymbol{r}_{obst} \in \mathcal{O}$. Therefore the safety constraints regarding the pair link-obstacle can be written as follows:

$$T_s\left(\boldsymbol{r}_{obst}^T \boldsymbol{E}_0 + \boldsymbol{E}_1\right)\dot{\boldsymbol{q}} \leq d, \quad \forall\, \boldsymbol{r}_{obst} \in \mathcal{O} \tag{25}$$

For linearity (and thus convexity) the aforementioned constraint (which actually consists of an infinite number of scalar

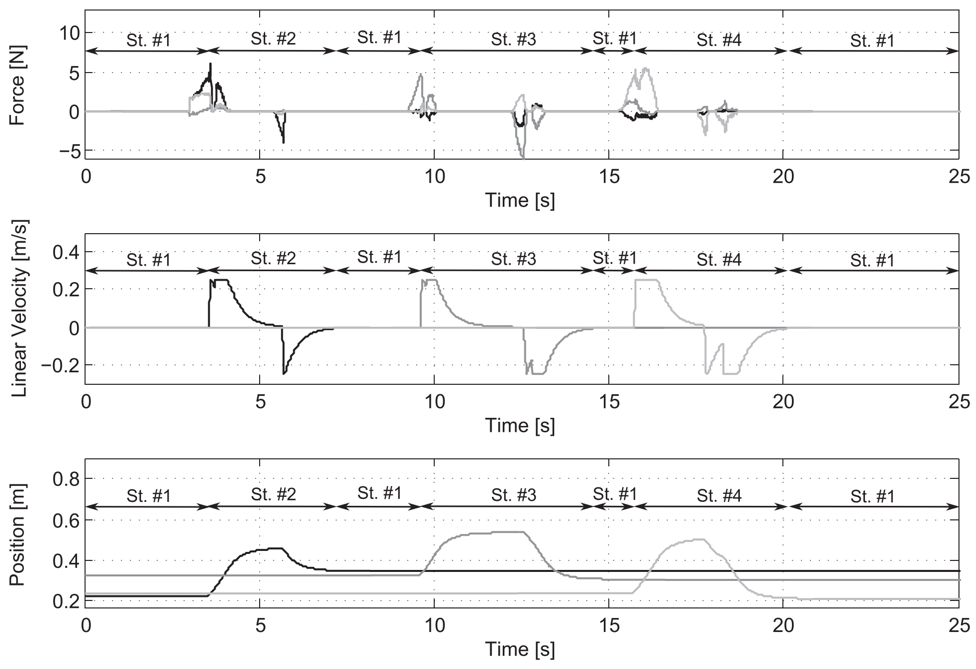


**Fig. 7.** Experiment #1. Top plot – estimation of external forces applied to the robot: $X$ component (solid black line), $Y$ component (solid dark grey line), $Z$ component (solid light grey line). Central plot – end-effector linear velocity: $X$ component (solid black line), $Y$ component (solid dark grey line), $Z$ component (solid light grey line). Bottom plot – end-effector Cartesian position: $X$ component (solid black line), $Y$ component (solid dark grey line), $Z$ component (solid light grey line). The corresponding FSM state is reported on top of the plots: State #1 – Stop, State #2 – Translation along $X$, State #3 – Translation along $Y$, State #4 – Translation along $Z$.

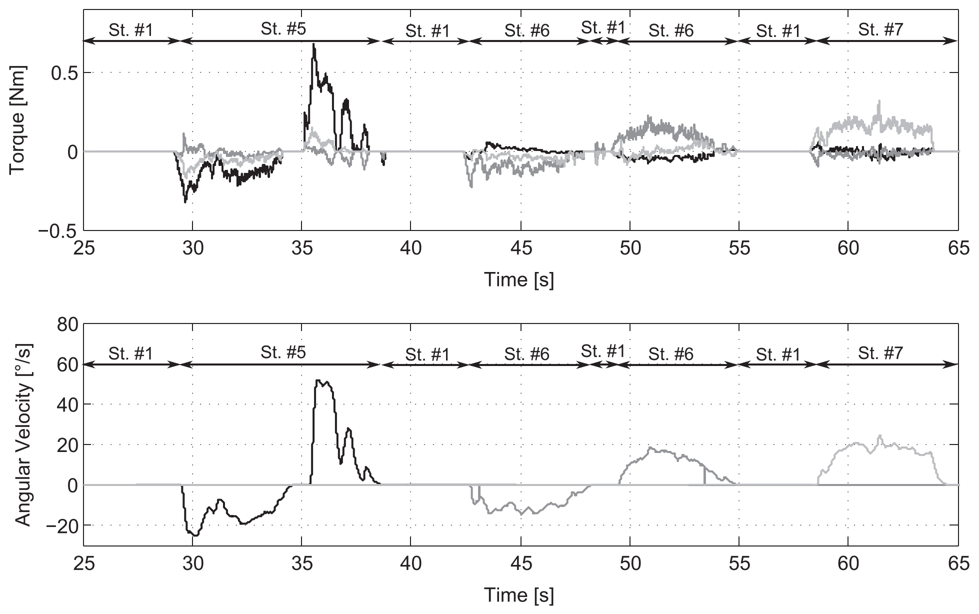


**Fig. 8.** Experiment #1. Top plot – estimation of external moments applied to the robot: $X$ component (solid black line), $Y$ component (solid dark grey line), $Z$ component (solid light grey line). Bottom plot – end-effector angular velocity: $X$ component (solid black line), $Y$ component (solid dark grey line), $Z$ component (solid light grey line). The corresponding FSM state is reported on top of the plots: State #1 – Stop, Stop, State #5 – Rotation around $X$, State #6 – Rotation around $Y$, State #7 – Rotation around $Z$.

inequalities) can be equivalently written in terms of the vertices (thus a limited number) of the polytope representing the obstacle $O$, hence $\forall\, \boldsymbol{r}_{obst} \in vert(O)$.

### 3.5. Lead-through programming algorithm and redundancy resolution

In the following, we present the whole algorithm developed for accurate lead-through programming, in presence of possible known obstacles. The algorithm consists of a quadratic programming (QP) problem which is solved at every time step in order to compute joint reference accelerations as control variables: $\boldsymbol{u}_k = \ddot{\boldsymbol{q}}_k^{ref}$. Then, position and velocity references ($\boldsymbol{q}_{k+1}^{ref}$ and $\dot{\boldsymbol{q}}_{k+1}^{ref}$, respectively) are updated on the basis of $\boldsymbol{u}_k$ and sent to the lower level axis controller. The goal of the QP problem is to allow best reference tracking of the outputs of the admittance filter $\sigma_{k+1}^{ref}$ and $\dot{\sigma}_{k+1}^{ref}$ in (10) or (13), whilst guaranteeing the satisfaction of joint position, velocity and accelerations limits, task space velocity limits, e.g. according to [32], and obstacle avoidance.

The complete QP problem is reported in (26).

$$\min_{\boldsymbol{u}_k, \ddot{\sigma}_k}\left( Q_v\left(\dot{\sigma}_{k+1} - \dot{\sigma}_{k+1}^{ref}\right)^2 + Q_p\left(\sigma_{k+1} - \sigma_{k+1}^{ref}\right)^2\right) \tag{26a}$$

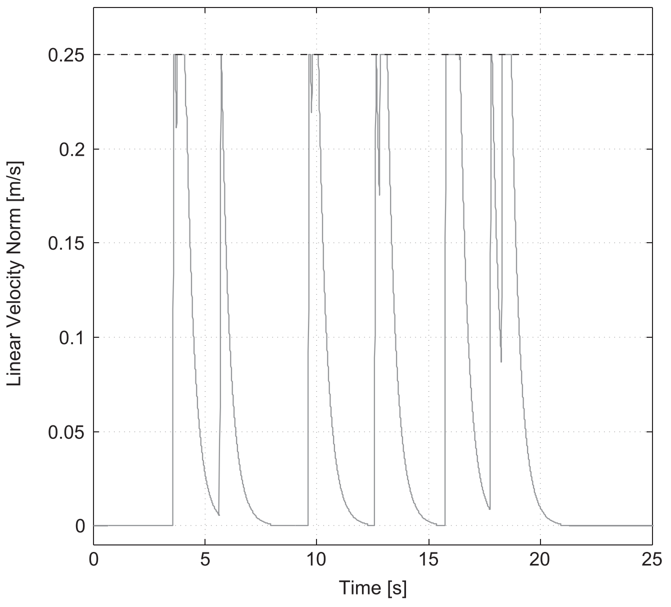


**Fig. 9.** Experiment #1: End-effector linear velocity norm (solid grey line) with respect to linear velocity upper bound (dashed black line).

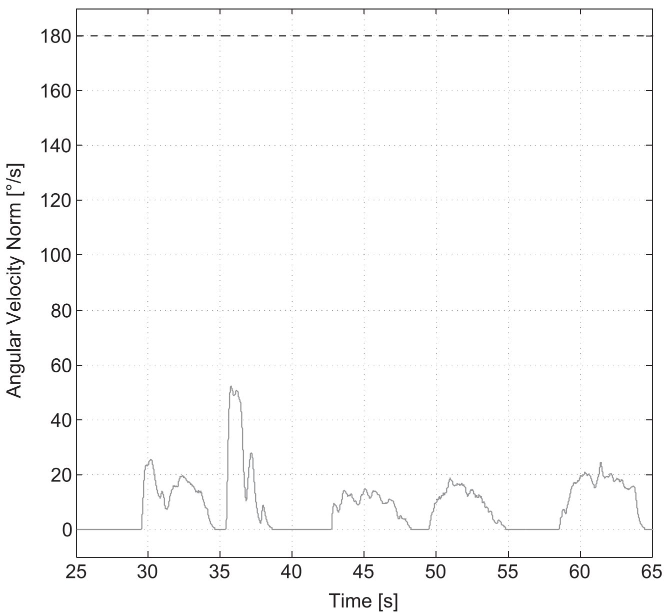


**Fig. 10.** Experiment #1: End-effector angular velocity norm (solid grey line) with respect to angular velocity upper bound (dashed black line).

$$\boldsymbol{J}_k \boldsymbol{u}_k + \dot{\boldsymbol{J}}_k \dot{\boldsymbol{q}}_k^{ref} = \boldsymbol{\Gamma}\ddot{\sigma}_k + \boldsymbol{K}_d \dot{\boldsymbol{e}}_k + \boldsymbol{K}_p \boldsymbol{e}_k \tag{26b}$$

$$\boldsymbol{u}_{inf} \leq \boldsymbol{u}_k \leq \boldsymbol{u}^{sup} \tag{26c}$$

$$-\dot{\sigma}^{max} \leq \dot{\sigma}_{k+1} \leq \dot{\sigma}^{max} \tag{26d}$$

$$-\ddot{\sigma}^{max} \leq \ddot{\sigma}_k \leq \ddot{\sigma}^{max} \tag{26e}$$

$$\boldsymbol{E}_k \boldsymbol{u}_k \leq \boldsymbol{f}_k \tag{26f}$$

where $\sigma_{k+1} = \sigma_k + \Delta t\dot{\sigma}_k + \frac{\Delta t^2}{2}\ddot{\sigma}_k$ and $\dot{\sigma}_{k+1} = \dot{\sigma}_k + \Delta t\acute{\sigma}_k$, $\boldsymbol{J}_k$ stands for $\boldsymbol{J}(\boldsymbol{q}_k^{ref})$, $\boldsymbol{e}_k = \sigma_k^{ref} - \sigma(\boldsymbol{q}_k^{ref})$ and $\dot{\boldsymbol{e}}_k = \dot{\sigma}_k^{ref} - \boldsymbol{J}_k \dot{\boldsymbol{q}}_k^{ref}$ are the position and velocity errors with respect to operational space references, while finally

$$\boldsymbol{\Gamma} = \begin{cases} \boldsymbol{n} & \text{State \#0, \#1, \#2, \#3, \#4} \\ \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix}^T & \text{otherwise} \end{cases}$$

The cost function (26a) is meant to guarantee best tracking performance of the output of the admittance filter (in terms of displacement and velocity). Constraints in (26b) represent a second order CLIK inverse kinematics algorithm, while those in (26c) implement joint position, velocity and acceleration limits, as well as (26d) and (26e) represent Cartesian acceleration and velocity limits. Finally, as discussed in Section 3.4, the inequalities in (26f) account for possible obstacles within the robot workspace to be avoided. Notice that the obstacle avoidance constraints have higher priority than the admittance rule in the cost function. Hence, in case the human operator tends to push the robot TCP against an obstacle, the QP algorithm will automatically decrease the velocity of the robot to avoid any possible collision between the robot and any workspace obstacle.

In addition, in case of kinematic redundancy of the robot, several solutions of the QP problem in (26) exist, differing in the null-space components of the optimal solution $\boldsymbol{u}_k$. For this reason, another optimization layer can be adopted to further optimize such components. In particular, following the approach in [33], the QP problem in (27) can be introduced. (26c) and (26f).

$$\min_{\boldsymbol{u}_k} \left( \frac{1}{2}\Delta t \boldsymbol{u}_k^T \boldsymbol{u}_k + \dot{\boldsymbol{q}}_k^T \boldsymbol{u}_k \right) \tag{27a}$$

$$\boldsymbol{J}_k \boldsymbol{u}_k = \boldsymbol{J}_k \boldsymbol{u}_k^0 \tag{27b}$$

$$\text{all the constraints in (26c) and (26f)} \tag{27c}$$

where $\boldsymbol{u}_k^0$ is any of the solutions of the QP problem in (26). The goal of this second QP problem is to preserve the optimality of the first problem, through the enforcement of constraint (27b), whilst selecting the smallest null space velocity compatible with all the other constraints.

Finally, position and velocity references at the next time step are obtained as follows:

$$\dot{\boldsymbol{q}}_{k+1}^{ref} = \dot{\boldsymbol{q}}_k^{ref} + \Delta t \boldsymbol{u}_k \tag{28}$$

$$\boldsymbol{q}_{k+1}^{ref} = \boldsymbol{q}_k^{ref} + \Delta t \dot{\boldsymbol{q}}_k^{ref} + \frac{\Delta t^2}{2}\boldsymbol{u}_k \tag{29}$$

and sent to the lower level axis controller.

Notice that, in case of saturation due to activation of constraints, the robot end-effector might not be able to follow the references computed by the admittance filter. For this reason, the output of the direct kinematics is sent back to the filter to avoid wind-up phenomena (see again Fig. 2). The state of the admittance filter is then updated as

$$\sigma_{k+1}^{ref} \leftarrow \boldsymbol{\Gamma}^T \sigma_{k+1} = \boldsymbol{\Gamma}^T \sigma\left(\boldsymbol{q}_{k+1}^{ref}\right) \tag{30}$$

$$\dot{\sigma}_{k+1}^{ref} \leftarrow \boldsymbol{\Gamma}^T \dot{\sigma}_{k+1} = \boldsymbol{\Gamma}^T \boldsymbol{J}\left(\boldsymbol{q}_{k+1}^{ref}\right)\dot{\boldsymbol{q}}_{k+1}^{ref} \tag{31}$$

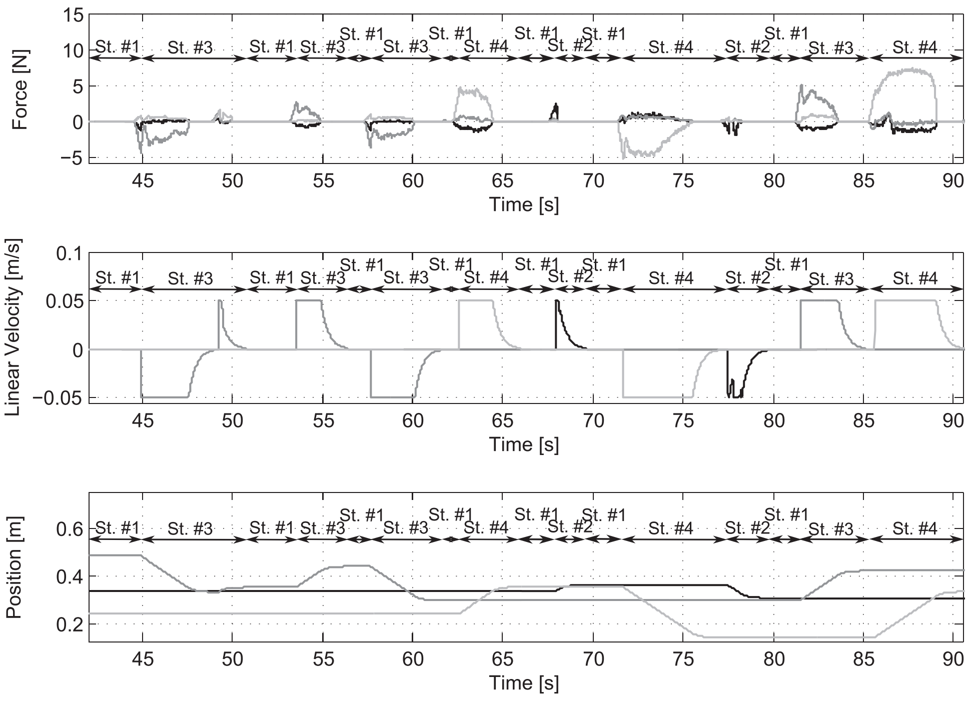


**Fig. 11.** Experiment #2. Top plot – estimation of external forces applied to the robot: *X* component (solid black line), *Y* component (solid dark grey line), *Z* component (solid light grey line). Central plot – end-effector linear velocity: *X* component (solid black line), *Y* component (solid dark grey line), *Z* component (solid light grey line). Bottom plot – end-effector Cartesian position: *X* component (solid black line), *Y* component (solid dark grey line), *Z* component (solid light grey line). The corresponding FSM state is reported on top of the plots: State #1 – Stop, State #2 – Translation along *X*, State #3 – Translation along *Y*, State #4 – Translation along *Z*.

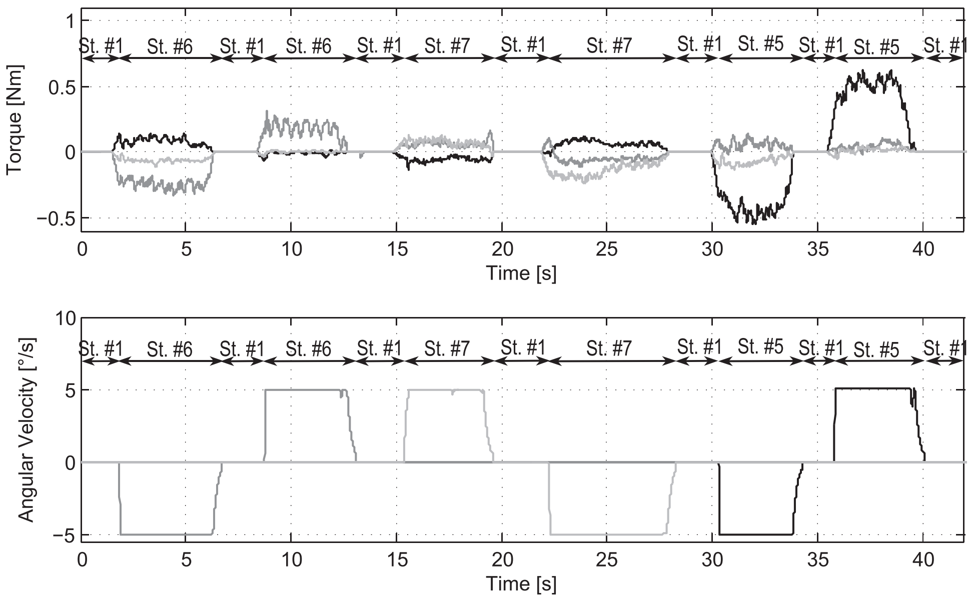


**Fig. 12.** Experiment #2. Top plot – estimation of external moments applied to the robot: *X* component (solid black line), *Y* component (solid dark grey line), *Z* component (solid light grey line). Bottom plot – end-effector angular velocity: *X* component (solid black line), *Y* component (solid dark grey line), *Z* component (solid light grey line). The corresponding FSM state is reported on top of the plots: State #1 – Stop, Stop, State #5 – Rotation around *X*, State #6 – Rotation around *Y*, State #7 – Rotation around *Z*.

## 4. Experiments

For the implementation of the proposed lead-through programming strategy, we here consider the 14-DOF dual-arm redundant robot prototype ABB FRIDA (see again Fig. 1), equipped with a control system based on ABB IRC5 industrial controller, located inside its torso.

An external PC is interfaced to the IRC5-based controller through an Ethernet-based interface (see [34] for details). This PC runs under Linux OS with the Xenomai patch, enabling a hard real-time system, and it acts as an external controller. Using the Ethernet-based interface, it is possible to develop a control algorithm within MATLAB Simulink on the external PC, and then compile it to get the executable code that runs and communicates in real-time with the IRC5-based controller at a frequency of 250 Hz. Full-duplex real-time communication allows the external controller to acquire data regarding the kinematic configuration of the manipulator (for logging purposes) and to override the reference signals to be sent to the low-level joint controllers (for control purposes).

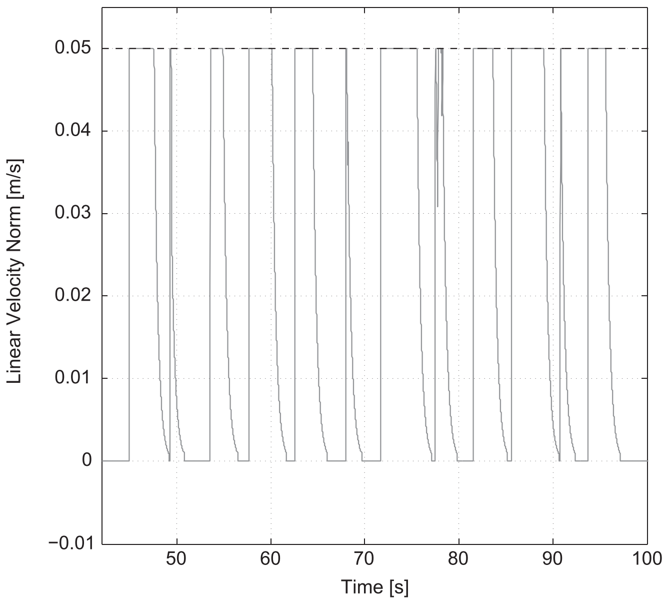


**Fig. 13.** Experiment #2: End-effector linear velocity norm (solid grey line) with respect to linear velocity upper bound (dashed black line).

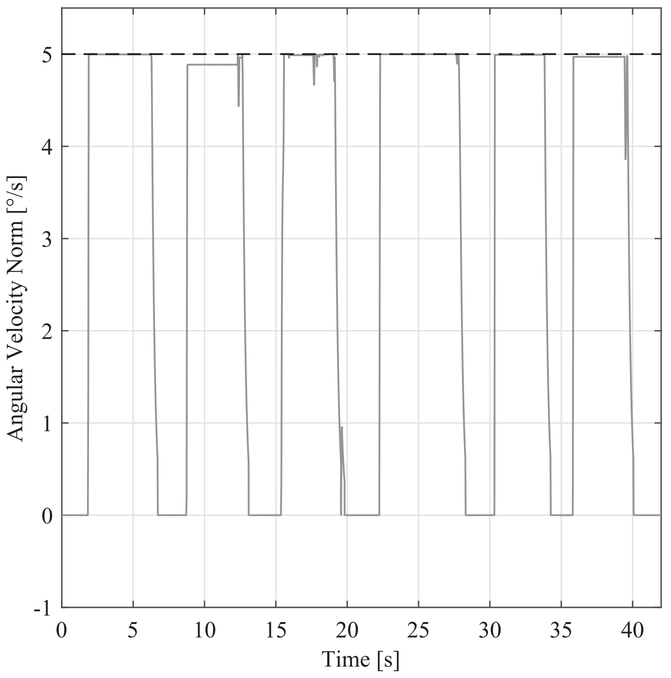


**Fig. 14.** Experiment #2: End-effector angular velocity norm (solid grey line) with respect to angular velocity upper bound (dashed black line).

In order to solve the QP problems (26) and (27), we use the “qpOASES” solver [35,36]. As far as the computational burden of the optimization stage is concerned, Fig. 6 shows the measured execution times of more than 25,000 calls to qpOASES organized in a histogram. On the other hand, Table 1 provides some more details regarding average, minimum and maximum execution time. Given these data, we can state that the QP solver can easily solve the optimization problems within the 4 ms cycle time of the real-time external controller.

In the following, we first provide details regarding the identification of the manipulator unknown dynamic parameters (along with results of a validation experiment of the dynamic model), then

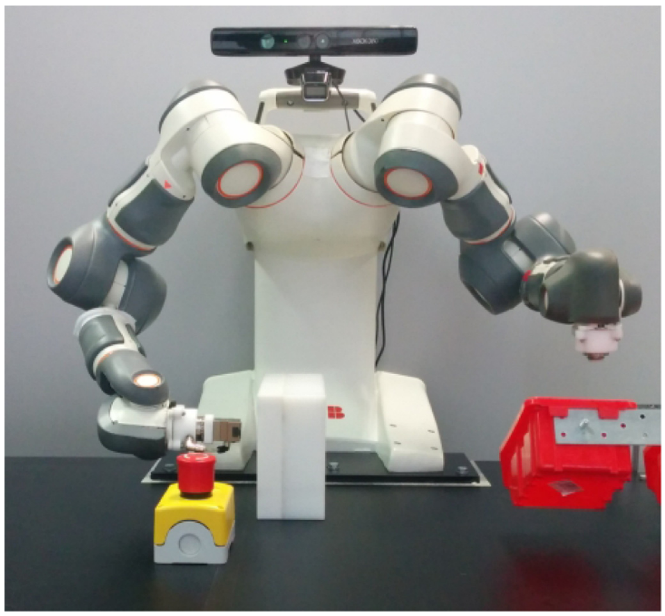


**Fig. 15.** The modified experimental setup for Experiment #3 with calibrated obstacle and goal object.

we present several validation experiments demonstrating the effectiveness of the proposed lead-through programming approach.

### 4.1. *Unknown dynamic parameter identification*

The inertial, centrifugal and gravitational terms of FRIDA's dynamic model have been assembled on the basis of the dynamic parameters provided by the manufacturer. In order to determine viscous and static friction coefficients, first an identification dataset has been acquired and then the identification procedure described in Section 2 has been performed.

Table 2 summarizes the results of a validation experiment performed on a validation dataset. Validation error statistics show the accuracy of the identified model and demonstrate that in this case it is absolutely reasonable to rely on manufacturer's datasheet for building the inertial, centrifugal and gravitational terms of the dynamic model.

### 4.2. *Experimental validation*

The lead-through robot programming approach described in the previous sections has been experimentally tested considering three different scenarios: lead-through without obstacles, lead-through without obstacles and reduced velocity limits, and finally lead-through in the presence of obstacles. In the following, each considered scenario is detailed and the corresponding experimental results are presented. The attached video integrates the description of the experiments and further documents their results.

#### 4.2.1. *Experiment #1: lead-through without obstacles*

In the first experiment, the robot is driven by the human worker without considering any known obstacle. The initial experimental setup is represented in Fig. 1.

At first the “orientation flag” is set to 0 and the human worker can modify only the end-effector Cartesian position. The estimated external forces applied by the human worker to the manipulator and the linear end-effector velocity components resulting from the lead-through programming algorithm are shown in Fig. 7.

From the collected data it is clear that the lead-through

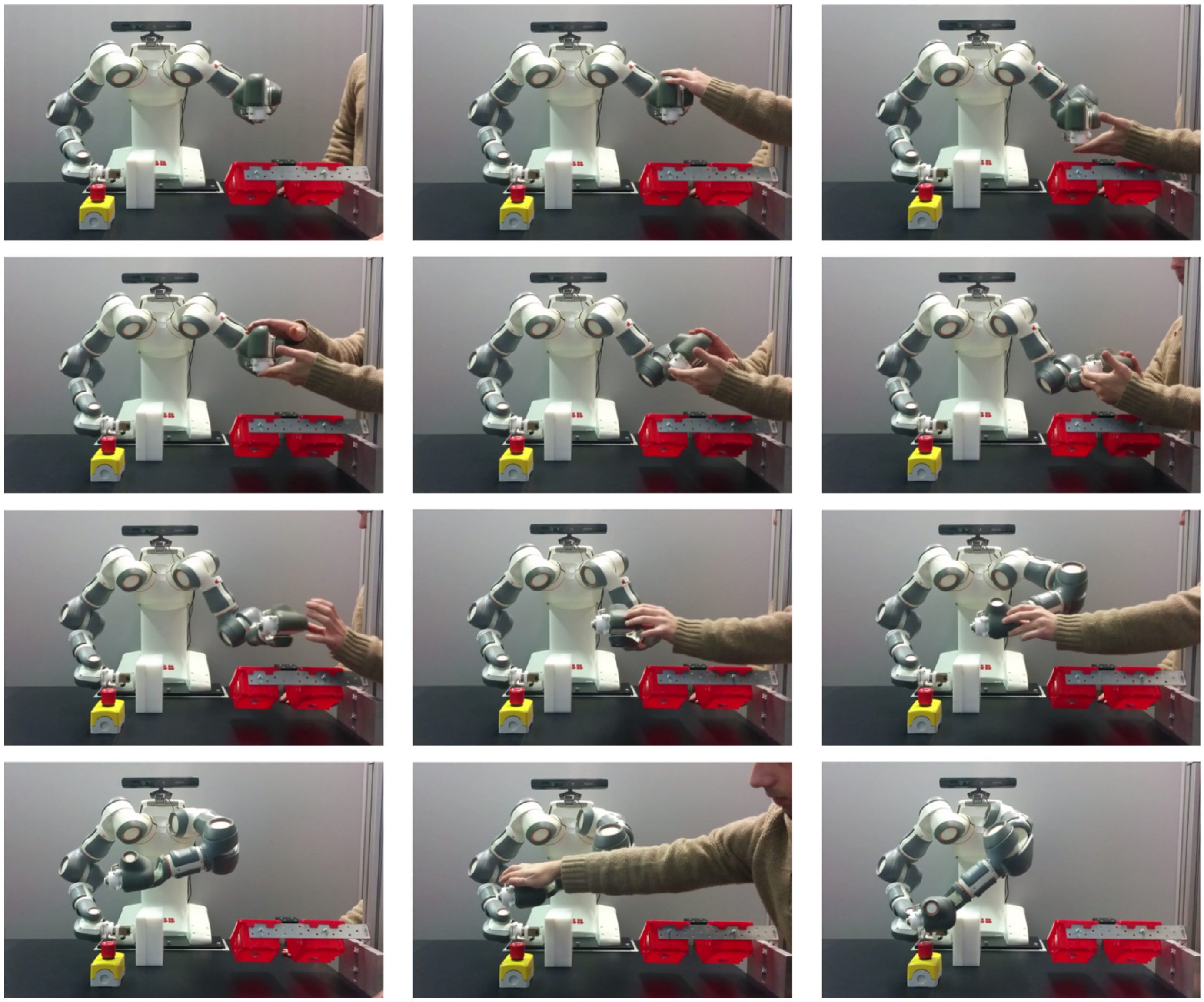


**Fig. 16.** Experiment #3: the manipulator is safely guided by the human operator towards its goal, without colliding with the obstacle (white box).

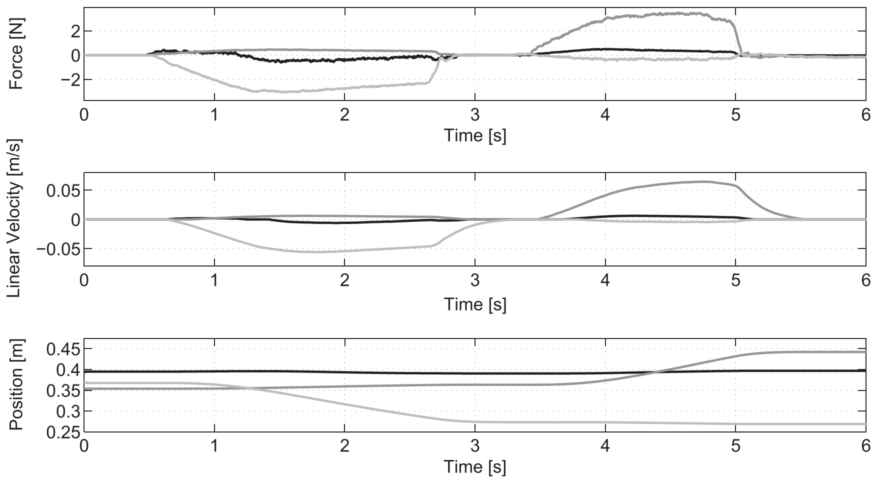


**Fig. 17.** Experiment #4. Top plot – estimation of external moments applied to the robot: *X* component (solid black line), *Y* component (solid dark grey line), *Z* component (solid light grey line). Bottom plot – end-effector angular velocity: *X* component (solid black line), *Y* component (solid dark grey line), *Z* component (solid light grey line).

programming algorithm correctly identifies the main component of the force applied by the human operator to the end-effector, allowing it to move only along the selected Cartesian direction.

Then, the human operator sets the orientation flag to 1 and starts modifying the end-effector orientation by applying moments. The estimation of these moments is shown in Fig. 8 along with the end-effector angular velocity components resulting from the lead-through programming algorithm.

Once again, experimental data prove that the algorithm successfully identifies the main component of the applied momentum, allowing the operator to enforce a rotation only along the selected Cartesian axis. Moreover these data demonstrate that our approach to lead-through programming provides a level of accuracy of the end-effector motion comparable to teach pendant-based programming while being definitely more user-friendly.

Finally, Fig. 9 shows that the end-effector linear velocity norm never overcomes the maximum limit of 0.250  m/s, while it can be seen in Fig. 10 that the end-effector angular velocity norm is always lower than the maximum limit of 180.00°/s.

### 4.2.2. *Experiment #2: lead-through without obstacles and reduced velocity limits*

A second experiment is performed with the human operator modifying the end-effector orientation at first and then varying the end-effector position. Differently from the previous experi-ment, the maximum end-effector linear and angular velocity limits are set to 0.05 m/s and 5.00°/s, respectively.

Estimated external forces are shown in Fig. 11, while the estimation of applied moments is depicted in Fig. 12. Not only Figs. 11 and 12 demonstrate that, once again, the algorithm successfully selects the right direction for both linear and rotational motion on the basis of the largest Cartesian component of the applied external forces and/or moments, but Figs. 13 and 14 clearly show that the end-effector linear and angular velocities saturate at the imposed limits, regardless of the magnitude of the applied force/momentum. This result demonstrates how safety aspects have been successfully incorporated in the proposed lead-through programming strategy by allowing the operator to set these velocity limits as configuration parameters for the algorithm.

### 4.2.3. *Experiment #3: lead-through in presence of obstacles*

In order to demonstrate the effectiveness of the proposed lead-through programming strategy in terms of obstacle avoidance, the experimental setup is modified by introducing an obstacle inside the manipulator workspace (the white box shown in Fig. 15). This obstacle is calibrated by measuring its position with respect to the robot base frame and by computing a simplified triangular mesh describing its geometry. In addition, a yellow and red stop button (see again Fig. 15) is introduced as a goal that the manipulator has to reach.

The effectiveness of the obstacle avoidance constraints is documented in the attached video and it is also demonstrated by several screenshots of the experiment shown in Fig. 16. While the human operator successfully guides the robot end-effector to the goal, the kinematic redundancy of the manipulator is exploited in order to comply with the obstacle avoidance constraints and to obtain a collision-free motion of the entire manipulator.

These results not only demonstrate that the programming algorithm can be easily configured in order to consider zero or more calibrated obstacles, but also show that kinematic redundancy is successfully exploited by the algorithm to ensure collision free motion of the entire manipulator during lead-through.

### 4.2.4. *Experiment #4: non-accurate lead-through*

In order to prove the accuracy of our approach with respect to standard LTP techniques, in the last experiment we disabled the voting system, the FSM and the optimization stage. The results shown in Fig. 17 demonstrate that, even though the force applied by the operator is always characterized by one Cartesian component significantly greater than the other ones, the physical interaction determines displacements of the end-effector position along all the Cartesian axes, thus producing a substantially less accurate motion.

As a matter of fact, accuracy issues necessarily arise when performing traditional LTP, since it is very difficult (if not impossible) for a human operator to apply a force/momentum exactly aligned with a specific Cartesian direction. For example, consider the first row in Fig. 16, which exemplifies the approach phase of a picking task (to be grasped from the red tray). Using a traditional LTP method, it would be difficult to maintain the alignment of the TCP with the object to be grasped. In turn, with the approach proposed in this paper, the human operator is only responsible for moving the robot towards the object, whilst the alignment is automatically guaranteed by the controller.

## 5. Conclusions and future developments

An approach to accurate lead-through robot programming has been presented and discussed in this paper. The described solution is based on the manipulator dynamic model and on proprioceptive measures typically available in an industrial manipulator (i.e. joint positions, joint velocities and motor torques) and does not rely on dedicated hardware. The proposed approach has been discussed and detailed in terms of identification of the dynamic model, development of a real-time external force estimation observer, identification of the largest Cartesian component of the force/momentum applied by the operator, usage of an implicit admittance control and introduction of an optimization framework in order to specify different types of constraints.

By simply overriding the joint position reference signals sent to the joint position control loop it is possible to implement the proposed lead-through programming solution without modifications to the closed industrial controller.

The effectiveness of the proposed approach is shown by an experimental validation and a video attachment.

Future developments are mainly represented by the possibility to apply this model based sensorless approach using more complex dynamic models that consider more sophisticated friction models and/or joint elasticity.

## Appendix A. Supplementary data

Supplementary data associated with this article can be found in the online version at: http://dx.doi.org/10.1016/j.rcim.2015.11.002.

## References

[1] Z. Pan, J. Polden, N. Larkin, S.V. Duin, J. Norrish, Recent progress on programming methods for industrial robots, Robot. Comput.: Integr. Manuf. 28 (2) (2012) 87–94.

[2] L. Bascetta, G. Ferretti, G. Magnani, P. Rocco, Walk-through programming for robotic manipulators based on admittance control, Robotica 31 (2013) 1143–1153.

[3] M. Jakopec, F. Rodriguez y Baena, S. Harris, P. Gomes, J. Cobb, B.L. Davies, The hands-on orthopaedic robot “acrobot” early clinical trials of total knee replacement surgery, IEEE Trans. Robot. Autom. 19 (5) (2003) 902–911.

[4] M. Jakopec, S.J. Harris, F. Rodriguez y Baena, P. Gomes, B.L. Davies, The acrobot system for total knee replacement, Ind. Robot.: Int. J. 30 (1) (2003) 61–66.

[5] P. Marayong, M. Li, A. Okamura, G. Hager, Spatial motion constraints: theory and demonstrations for robot guidance using virtual fixtures, In: Proceedings of the IEEE International Conference on Robotics and Automation, 2003, ICRA '03, vol. 2, pp. 1954–1959.

[6] M. Li, M. Ishii, R. Taylor, Spatial motion constraints using virtual fixtures generated by anatomy, IEEE Trans. Robot. 23 (1) (2007) 4–19.

[7] K. Salisbury, W. Townsend, B. Ebrman, D. DiPietro, Preliminary design of a whole-arm manipulation system (wams), In: Proceedings of the IEEE International Conference on Robotics and Automation, vol. 1, 1988, pp. 254–260.

[8] G. Hirzinger, A. Albu-Schäffer, M. Hahnle, I. Schaefer, N. Sporer, On a new generation of torque controlled light-weight robots, In: Proceedings of the IEEE International Conference on Robotics and Automation, ICRA, vol. 4, 2001, pp. 3356–3363.

[9] A. De Luca, W. Book, Robots with Flexible Elements, Springer, Berlin, Heidelberg, 2008.

[10] C. Ott, C. Ott, A. Kugi, G. Hirzinger, On the passivity-based impedance control of flexible joint robots, IEEE Trans. Robot. 24 (2) (2008) 416–429.

[11] R. Schiavi, G. Grioli, S. Sen, A. Bicchi, Vsa-ii: a novel prototype of variable stiffness actuator for safe and performing robots interacting with humans, In: IEEE International Conference on Robotics and Automation, 2008, ICRA 2008, pp. 2171–2176.

[12] K. Suita, Y. Yamada, N. Tsuchida, K. Imai, H. Ikeda, N. Sugimoto, A failure-to-safety "kyozon" system with simple contact detection and stop capabilities for safe human–autonomous robot coexistence, In: Proceedings of the IEEE International Conference on Robotics and Automation, 1995, vol. 3, pp. 3089–3096.

[13] A.R. Products, Collision Detection RobotWare option – Version 2.1, ABB, SE-72168 Vasteras, Sweden, 2008, PR10044en.

[14] M. Geravand, F. Flacco, A. De Luca, Human–robot physical interaction and collaboration using an industrial robot with a closed control architecture, In: IEEE International Conference on Robotics and Automation (ICRA), 2013, pp. 4000–4007.

[15] A. De Luca, R. Mattone, Sensorless robot collision detection and hybrid force/motion control, In: Proceedings of the 2005 IEEE International Conference on Robotics and Automation, ICRA 2005, pp. 999–1004.

[16] A. De Luca, A. Albu-Schäffer, S. Haddadin, G. Hirzinger, Collision detection and safe reaction with the dlr-iii lightweight manipulator arm, In: 2006 IEEE/RSJ International Conference on Intelligent Robots and Systems, 2006, pp. 1623–1630.

[17] S. Haddadin, A. Albu-Schäffer, A. De Luca, G. Hirzinger, Collision detection and reaction: a contribution to safe physical human–robot interaction, In: IEEE/RSJ International Conference on Intelligent Robots and Systems, 2008, IROS 2008, pp. 3356–3363.

[18] M. Erden, T. Tomiyama, Human-intent detection and physically interactive control of a robot without force sensors, IEEE Trans. Robot. 26 (2) (2010) 370–382.

[19] G. Magnani, P. Rocco, L. Bascetta, A. Rusconi, On the use of torque disturbance observers in 2-mass systems with application to a robotic joint, In: 2013 IEEE International Conference on Mechatronics (ICM), 2013, pp. 798–803.

[20] E. Magrini, F. Flacco, A. De Luca, Estimation of contact forces using a virtual force sensor, In: 2014 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2014), 2014, pp. 2126–2133.

[21] H. Sadeghian, M. Keshmiri, L. Villani, B. Siciliano, Null-space impedance control with disturbance observer, In: 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2012, pp. 2795–2800.

[22] H. Sadeghian, L. Villani, M. Keshmiri, B. Siciliano, Task-space control of robot manipulators with null-space compliance, IEEE Trans. Robot. PP (99) (2013) 1–14.

[23] M. Linderoth, A. Stolt, A. Robertsson, R. Johansson, Robotic force estimation using motor torques and modeling of low velocity friction disturbances, In: 2013 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2013, pp. 3550–3556.

[24] A. Stolt, M. Linderoth, A. Robertsson, R. Johansson, Force controlled robotic assembly without a force sensor, in: 2012 IEEE International Conference on Robotics and Automation (ICRA), 2012, pp. 1538–1543.

[25] A. Stolt, M. Linderoth, A. Robertsson, R. Johansson, Robotic assembly of emergency stop buttons, in: 2013 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2013, pp. 2081–2081.

[26] A. Zanchettin, P. Rocco, Path-consistent safety in mixed human-robot collaborative manufacturing environments, In: 2013 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2013, pp. 1131–1136.

[27] A. Shiriaev, A. Robertsson, R. Johansson, Friction compensation for passive systems based on the LuGre model, in: Proceedings of the 2nd IFAC Workshop on Lagrangian and Hamiltonian Methods for Nonlinear Control, Seville, Spain, 2003, pp. 183–188.

[28] A. Stolt, A. Robertsson, R. Johansson, Robotic force estimation using dithering to decrease the low velocity friction uncertainties, In: 2015 IEEE International Conference on Robotics and Automation, Seattle, WA, USA, 2015.

[29] G. Ferretti, G. Magnani, P. Rocco, Impedance control for elastic joints industrial manipulators, IEEE Trans. Robot. Autom. 20 (3) (2004) 488–498.

[30] ANSI/RIA R15 06, Safety Requirements for Industrial Robots and Robot Systems, 1999.

[31] E. Gilbert, D. Johnson, S. Keerthi, A fast procedure for computing the distance between complex objects in three-dimensional space, IEEE J. Robot. Autom. 4 (2) (1988) 193–203.

[32] ISO 15066, Robots and Robotic Devices—Safety Requirements for Industrial Robots—Collaborative Operation, ISO, Geneva, Switzerland, 2013.

[33] O. Kanoun, F. Lamiraux, P.-B. Wieber, Kinematic control of redundant manipulators generalizing the task-priority framework to inequality task, IEEE Trans. Robot. 27 (4) (2011) 785–792.

[34] A. Blomdell, I. Dressler, K. Nilsson, A. Robertsson, Flexible application development and high-performance motion control based on external sensing and reconfiguration of ABB industrial robot controllers, In: Proceedings of ICRA 2010 Workshop on Innovative Robot Control Architectures for Demanding (Research) Applications, Anchorage, AK, 2010.

[35] H. Ferreau, H. Bock, M. Diehl, An online active set strategy to overcome the limitations of explicit mpc, Int. J. Robust Nonlinear Control 18 (8) (2008) 816–830.

[36] H. Ferreau, C. Kirches, A. Potschka, H. Bock, M. Diehl, qpOASES a parametric active-set algorithm for quadratic programming, Math. Program. Comput. 6 (4) (2014) 327–363.