# Adaptive Constrained Kinematic Control using Partial or Complete Task-Space Measurements

Murilo M. Marinho and Bruno V. Adorno

***Abstract*— Recent advancements in constrained kinematic control make it an attractive strategy for controlling robots with arbitrary geometry in challenging tasks. Most current works assume that the robot kinematic model is precise enough for the task at hand. However, with increasing demands and safety requirements in robotic applications, there is a need for a controller that compensates online for kinematic inaccuracies. We propose an adaptive constrained kinematic control strategy based on quadratic programming, which uses partial or complete task-space measurements to compensate online for calibration errors. Our method is validated in experiments that show increased accuracy and safety compared to a state-of-the-art kinematic control strategy.**

## I. Introduction

Kinematic control has been applied effectively to a myriad of tasks that use velocity-actuated robots with distinct geometry, such as manipulator robots, mobile robots, and humanoid robots.

Kinematic control strategies use a model derived from the geometric parameters of the robotic system [1]. The accuracy[1] of the control strategy is directly related to the accuracy of the robot's geometric parameters, such as link length for manipulator robots, wheel radius for mobile robots, and so on. Moreover, in applications that require cooperation between robots and/or between robots and humans, the geometric parameters can also include the relative pose between the reference frame of each robot and other entities in the workspace.

In applications in which accuracy is not a major requirement, using the parameters defined in the robot's design plans (schematics, computer-assisted design files, etc.) is accurate enough. For some applications that require a finer degree of accuracy, it is common practice to perform a calibration process before executing the task to obtain a more reliable estimate of the robot parameters [2]–[5]. A calibration process that happens before the task execution is hereby called an *offline* calibration strategy.

This work was supported in part by JSPS KAKENHI Grant Number JP19K14935 and in part by the ImPACT Program of Council for Science, Technology and Innovation Grant Number 2015-PM15-11-01. *(Corresponding author:* Murilo M. Marinho.)

Murilo M. Marinho is with the Department of Mechanical Engineering, the University of Tokyo, Tokyo, Japan. Email:murilo@g.ecc.u-tokyo.ac.jp.

Bruno V. Adorno is with the Department of Electrical and Electronic Engineering, The University of Manchester, Sackville Street, Manchester, M13 9PL, United Kingdom . Email: bruno.adorno@manchester.ac.uk.

[1]Accuracy is a measure of how close a robot is able to move its end effector to a desired point in its task space.

In this work, our main motivations are applications in which offline calibration is impractical and/or insufficient, and task-space constraints are necessary. A few examples include medical applications in constrained workspaces, in which the robot might have to be repositioned to be used in different steps of the surgery [6]; reconfigurable robots, for which the kinematics may change according to novel configurations, especially when the coupling between different robot parts within a new configuration is not precise enough; or even assistant mobile manipulators handling different loosely grasped tools. In those cases, it is not practical (or even impossible) to temporarily stop the task to perform a time-consuming calibration procedure.

Moreover, in realistic applications, sensors that provide *complete* task-space measurements might not be available, especially when the workspace is constrained. For instance, in robot-assisted surgery for endonasal [6] or neonate procedures [7], the constrained workspace prevents the use of extra sensors. Also, having a reliable estimate of the robotic instruments' tip pose just from the endoscopic camera is challenging. A more realistic approach would be to use the information of the instruments' shaft centerline [8], [9].

To further enable using robots in realistic and challenging scenarios, our interest lies in integrating an *online* calibration strategy with constrained kinematic control to make the best of either *partial* or complete task-space measurements. In this sense, online refers to calibration being performed *at the same time* as the task is being executed by the robot, resulting in an adaptive controller.

### A. Related works

Constrained kinematic (and dynamic) control has received a lot of attention and has been extended to take into account task inequalities [10], provide a fast hierarchical solution [11], prevent collision with static and moving obstacles [12], and prevent self collisions [13]. All these works suppose that the robot model is precise enough after an offline calibration is performed.

Offline calibration strategies rely on having high-precision measuring equipment in a highly controlled setting [2]–[5]. In those approaches, the robotic system is placed in the calibration setup that might be composed of cameras and markers, coordinate-measurement machines, etc. Data are obtained from moving the robot around the workspace and measuring end-effector poses by using markers with known geometry or with respect to reference objects in the workspace. The kinematic parameters are obtained from an optimization algorithm that minimizes the error between

the measured data (e.g., pose, position, orientation) and the corresponding data obtained from the estimated parameters.

Inspired by those offline techniques, recent works [14], [15] have integrated high-precision sensors in the robot's workspace in industrial settings, which allows for the periodic update of the robot parameters. However, those works do not provide updates while the task is being performed.

Recent works have explored the optimal path to calibrate the robot's dynamic and geometric parameters in an offline fashion [16], [17]. Those works use constrained quadratic optimization strategies to minimize the time required for offline calibration by proving a "sufficiently rich" trajectory [18], [19].

To the best of our knowledge, one of the best techniques to the target application of this work is adaptive control because it compensates for inaccuracies in the robot model while the task is performed. For instance, the dynamic parameters [20], [21] and some kinematic parameters [22] of robots can be estimated and adjusted online. However, there are two major drawbacks of most existing adaptive controllers in this context: first, with very few exceptions, most of them require linearity on the kinematic parameters [23] because they rely on the so-called *parameter regression matrix*; second, they cannot consider task-space inequality constraints. Most of the research making use of the parameter regression matrix is in the context of grasping objects with uncertain weight using dynamic control. In that case, there are efficient algorithms to obtain the parameter regression matrix when, for instance, only the inertial parameters are to be adapted [24]. A parameter regressor can be found algebraically when the system is simple and there are few parameters to be adapted [19], [22], [25]. Based on the observations that it is not possible, in general, to find linearity on the kinematic parameters, Marcucci et al. [26] proposed a technique to factorize the robot kinematics into a regressor matrix and a vector of nonlinear functions of the uncertain parameters. Then an adaptive control law that updates the estimate of these nonlinear functions of the unknown parameters is proposed. Nonetheless, in addition to assuming that the model is linearly parameterizable and that the robot Jacobian is invertible, they do not account for constraints, and the controller requires complete task-space measurements. Lastly, the proof of closed-loop stability in existing works on adaptive control is not trivially extensible to constrained kinematic control, especially when inequality constraints are considered.

Data-driven approaches such as reinforcement learning [27] aim to find a suitable reward function to perform a task and has been idealized to take into account autonomous systems with arbitrary geometry. The technique sought in this work would be complementary to research efforts in reinforcement learning. This is because we aim for an adaptive constrained controller that guarantees safety and adapts the robot's uncertain model online according to available measurements. Therefore, it could also be used to improve sim-to-real transfer [28], [29] by adapting the learned robot's model online. Similarly, many trajectory planning approaches [30], [31] could directly benefit from such controller for the same reasons. Part of the planning can be performed offline and the constrained adaptive constrained controller can allow the transfer of this planning to real scenarios.

Visual servoing is a technique that relies on cameras to control the motion of robotic systems. Most classic techniques in visual servoing require the system's Jacobian to be invertible (at least those with stability proof), need a continuous stream of reliable camera readings, and do not address workspace constraints. Recent approaches [32] take advantage of data-driven algorithms to overcome one or more of those limitations. For instance, recent works address visual servoing under temporary occlusions [33] or tackle configuration-space constraints by using a data-driven strategy to learn the robot's model [34]. Related data-driven approaches allow for impressive performance in robot grasping in low-risk and low-accuracy scenarios [35]. In contrast with these existing approaches in visual servoing, we are interested in making use of *any* sensor (not only cameras) that can provide a possibly discontinuous stream of data to update the robot model in scenarios that might demand high-accuracy, with nonlinear workspace constraints, and without imposing any requirement on the Jacobian, such as invertibility.

Also, it is important to mention that this work does not aim to address hand/eye calibration of cameras, which has been extensively studied in the literature [8], [36], [37]. In the current study, we address task-space measurements only, and the estimation of a camera's intrinsic and extrinsic parameters are out of this paper's scope.

### *B. Statement of contributions*

In this work, we build upon our constrained kinematic controller [12] and propose an adaptive constrained kinematic control strategy that makes use of partial or complete task-space measurements while taking into account both equality and inequality constraints that are linear in the control inputs. To account for nonlinear constraints in both task-space variables and robot configurations, we extend the vector field inequalities (VFIs) [12] framework to our adaptive controller, such that those nonlinear constraints are transformed into linear differential inequalities in the control inputs.

Differently from existing approaches, our strategy does not demand a parameter regression matrix, requiring only differentiability of the forward kinematics with respect to the estimated parameters. Using Lyapunov stability theory, we show that the closed-loop system is stable in the sense that the error between the estimated end-effector pose and the desired pose is always non-increasing. We perform two experiments. In the first experiment, we perform a thorough experimental evaluation of the proposed methodology, in which we vary the amount of information available about the task-space. In the second experiment, we evaluate the controller in a scenario requiring multiple nonlinear task-space constraints.

## II. Problem Definition

Before describing the proposed adaptive control strategy, this section introduces the online calibration problem with (partial) measurements, in general terms. Also, we present the mathematical notation used in this paper, which is summarized in Table I.

TABLE I

Mathematical notation.

| Notation | Definition |
|---|---|
| $\mathcal{Q}_\mathrm{r} \subseteq \mathcal{Q} \subseteq \mathbb{R}^n$ | Restricted configuration space and configuration space of $n$ dimensions. |
| $\mathcal{A}_\mathrm{r} \subseteq \mathcal{A} \subseteq \mathbb{R}^p$ | Restricted parameter space and parameter space of $p$ dimensions. |
| $\mathscr{T}_\mathrm{r} \subseteq \mathscr{T} \subseteq \mathbb{R}^m$ | Restricted task-space and task-space of $m$ dimensions. |
| $\mathcal{Y} \subseteq \mathbb{R}^r$ | Measurement space of $r$ dimensions, with $r \leq m$. |
| $\overline{\mathcal{Y}} \subseteq \mathbb{R}^{\bar{r}}$ | Space of unmeasured variables of $\bar{r}$ dimensions. |
| $s$ | Number of constraints dependent on both the control inputs and the adaptation signal. |
| $s_q$ | Number of constraints dependent only on the control inputs. |
| $s_{\hat{a}}$ | Number of constraints dependent only on the adaptation signal. |
| $\boldsymbol{x} \in \mathscr{T}, \boldsymbol{y} \in \mathcal{Y}$ | Real task-space and measurement-space vectors, respectively. |
| $\hat{\boldsymbol{x}} \in \mathscr{T}, \hat{\boldsymbol{y}} \in \mathcal{Y}$ | Estimated task-space and estimated measurement-space vectors, respectively. |
| $\dot{\boldsymbol{x}} \in \mathscr{T}$ | Error between the estimated ($\hat{\boldsymbol{x}}$) and desired ($\boldsymbol{x}_d$) task-space vectors. |
| $\dot{\boldsymbol{y}} \in \mathcal{Y}$ | Error between the estimated ($\hat{\boldsymbol{y}}$) and real ($\boldsymbol{y}$) measure-space vectors. |

### A. Uncertain kinematic model

Consider a velocity (or position) actuated robotic system[2] with $n$ degrees-of-freedom (DoF) and configuration vector given by $\boldsymbol{q} \triangleq \boldsymbol{q}(t) \in \mathcal{Q}_\mathrm{r}$, in which the restricted configuration space is given by

$$\mathcal{Q}_\mathrm{r} \triangleq \{\boldsymbol{q} \in \mathcal{Q} : \boldsymbol{q}_\mathrm{min} \preceq \boldsymbol{q} \preceq \boldsymbol{q}_\mathrm{max} \text{ and } \boldsymbol{q}_\mathrm{min}, \boldsymbol{q}_\mathrm{max} \in \mathcal{Q} \subseteq \mathbb{R}^n\}.$$

Let the $p$ *real* parameters of the robot kinematic model, which are impossible to measure directly, be $\boldsymbol{a} \in \mathcal{A}_\mathrm{r}$, where

$$\mathcal{A}_\mathrm{r} \triangleq \{\boldsymbol{a} \in \mathcal{A} : \boldsymbol{a}_\mathrm{min} \preceq \boldsymbol{a} \preceq \boldsymbol{a}_\mathrm{max} \text{ and } \boldsymbol{a}_\mathrm{min}, \boldsymbol{a}_\mathrm{max} \in \mathcal{A} \subseteq \mathbb{R}^p\}.$$

If all parameters and joint values were perfectly known, the forward kinematics model (FKM) could be used to obtain the *real* task-space variables, $\boldsymbol{x} \triangleq \boldsymbol{x}(\boldsymbol{q}, \boldsymbol{a}) \in \mathscr{T} \subseteq \mathbb{R}^m$. The FKM is the nonlinear mapping

$$\boldsymbol{f} : \mathcal{Q} \times \mathcal{A} \to \mathscr{T}, \qquad (\boldsymbol{q}, \boldsymbol{a}) \mapsto \boldsymbol{x}, \tag{1}$$

and the restricted task space is

$$\mathscr{T}_\mathrm{r} \triangleq \{\boldsymbol{f}(\boldsymbol{q}, \boldsymbol{a}) \in \mathscr{T} : \boldsymbol{q} \in \mathcal{Q}_\mathrm{r},\, \boldsymbol{a} \in \mathcal{A}_\mathrm{r}\} \subseteq \mathscr{T}.$$

Let the *estimated* parameters be $\hat{\boldsymbol{a}}(t) \triangleq \hat{\boldsymbol{a}} \in \mathcal{A}_\mathrm{r}$. The *estimated* task-space vector $\hat{\boldsymbol{x}} \triangleq \hat{\boldsymbol{x}}(\boldsymbol{q}, \hat{\boldsymbol{a}}) \in \mathscr{T}_\mathrm{r}$ is obtained as

$$\hat{\boldsymbol{x}} \triangleq \boldsymbol{f}(\boldsymbol{q}, \hat{\boldsymbol{a}}). \tag{2}$$

The first time-derivative of (2) is the *estimated* differential forward kinematic model (DFKM)

$$\dot{\hat{\boldsymbol{x}}} = \underbrace{\frac{\partial \boldsymbol{f}(\boldsymbol{q}, \hat{\boldsymbol{a}})}{\partial \boldsymbol{q}}}_{\boldsymbol{J}_{\hat{x},q}} \dot{\boldsymbol{q}} + \underbrace{\frac{\partial \boldsymbol{f}(\boldsymbol{q}, \hat{\boldsymbol{a}})}{\partial \hat{\boldsymbol{a}}}}_{\boldsymbol{J}_{\hat{x},\hat{a}}} \dot{\hat{\boldsymbol{a}}}, \tag{3}$$

where $\boldsymbol{J}_{\hat{x},q} \triangleq \boldsymbol{J}_{\hat{x},q}(\boldsymbol{q}, \hat{\boldsymbol{a}}) \in \mathbb{R}^{m \times n}$ is the estimated task Jacobian and $\boldsymbol{J}_{\hat{x},\hat{a}} \triangleq \boldsymbol{J}_{\hat{x},\hat{a}}(\boldsymbol{q}, \hat{\boldsymbol{a}}) \in \mathbb{R}^{m \times p}$ is the estimated parametric Jacobian.

Let us consider a measurement system capable of measuring an $r$-dimensional subset of the real task-space variables given by $\boldsymbol{y} \triangleq \boldsymbol{y}(t) \in \mathcal{Y} \subseteq \mathbb{R}^r$, and the $\overline{r}$-unmeasured dimensions compose the space of unmeasured variables[3] $\overline{\mathcal{Y}} \subseteq \mathbb{R}^{\bar{r}}$. In addition, let the estimated measure-space FKM be given by the surjection

$$\boldsymbol{g} : \mathcal{Q}_\mathrm{r} \times \mathcal{A}_\mathrm{r} \twoheadrightarrow \mathcal{Y}, \qquad (\boldsymbol{q}, \hat{\boldsymbol{a}}) \mapsto \hat{\boldsymbol{y}}, \tag{4}$$

with DFKM

$$\dot{\hat{\boldsymbol{y}}} = \underbrace{\frac{\partial \boldsymbol{g}(\boldsymbol{q}, \hat{\boldsymbol{a}})}{\partial \boldsymbol{q}}}_{\boldsymbol{J}_{\hat{y},q}} \dot{\boldsymbol{q}} + \underbrace{\frac{\partial \boldsymbol{g}(\boldsymbol{q}, \hat{\boldsymbol{a}})}{\partial \hat{\boldsymbol{a}}}}_{\boldsymbol{J}_{\hat{y},\hat{a}}} \dot{\hat{\boldsymbol{a}}}, \tag{5}$$

where $\boldsymbol{J}_{\hat{y},q} \triangleq \boldsymbol{J}_{\hat{y},q}(\boldsymbol{q}, \hat{\boldsymbol{a}}) \in \mathbb{R}^{r \times n}$ is the estimated measure-space task Jacobian and $\boldsymbol{J}_{\hat{y},a} \triangleq \boldsymbol{J}_{\hat{y},a}(\boldsymbol{q}, \hat{\boldsymbol{a}}) \in \mathbb{R}^{r \times p}$ is the estimated measure-space parametric Jacobian.

### B. Problem statement and main assumptions

Given an external measurement system that provides partial or complete measurements $\boldsymbol{y} \in \mathcal{Y}$, drive the real task-space variable $\boldsymbol{x}(\boldsymbol{q}(t), \boldsymbol{a}(t)) \in \mathscr{T}_\mathrm{r}$ as close as possible to a constant desired value $\boldsymbol{x}_d \in \mathscr{T}$.

We assume that:

1) Perfect measurements are obtained in the task space and the configuration space. Any sensor that provides a partial or complete measurement of the task-space can be used.
2) All constraints are represented by differentiable functions and no constraint is violated at $t = 0$.
3) The control input $\boldsymbol{u}_q = \dot{\boldsymbol{q}}(t) = \boldsymbol{0}$ is a feasible solution for all $t \geq 0$, which means that the robot is always able to stop.

## III. Proposed control strategy

To control the robotic system while updating the robot kinematic parameters, we propose a control strategy based on two independent quadratic[4] programming (QP) problems that are solved either in cascade or simultaneously to generate each instantaneous control input.

First, a task-space control law uses the estimated parameters, $\hat{\boldsymbol{a}}$, and the desired task-space reference, $\boldsymbol{x}_d$, to calculate the optimal joint-velocity control input $\dot{\boldsymbol{q}} \triangleq \boldsymbol{u}_q$, which is sent to the robotic system. Then, an adaptive law generates the optimal adaptation signal $\dot{\hat{\boldsymbol{a}}} \triangleq \boldsymbol{u}_{\hat{a}}$ using the current task-space measurement $\boldsymbol{y}$ and robot configuration $\boldsymbol{q}$ to update the estimated kinematic parameters. The process repeats until $\hat{\boldsymbol{x}}(\boldsymbol{q}(t), \hat{\boldsymbol{a}}(t))$ becomes as close as possible to $\boldsymbol{x}_d$. The block diagram of the control strategy is shown in Fig. 1.

[2]The robotic system can be comprised of any number of individual robots.

[3]When both $\mathcal{Y}$ and $\overline{\mathcal{Y}}$ are parameterized with minimal representations, $\bar{r} = m - r$. However, if nonminimal representations are used, such that the number of independent coordinates in $\mathscr{T}$, $\mathcal{Y}$, and $\overline{\mathcal{Y}}$ are $m_\mathrm{ind} < m$, $r_\mathrm{ind} < r$, and $\bar{r}_\mathrm{ind} < \bar{r}$, respectively, then $\bar{r}_\mathrm{ind} = m_\mathrm{ind} - r_\mathrm{ind}$ but $\bar{r} \neq m - r$.

[4]Linear-quadratic optimization problems with inequality constraints cannot be solved analytically and are solved with numerical methods.

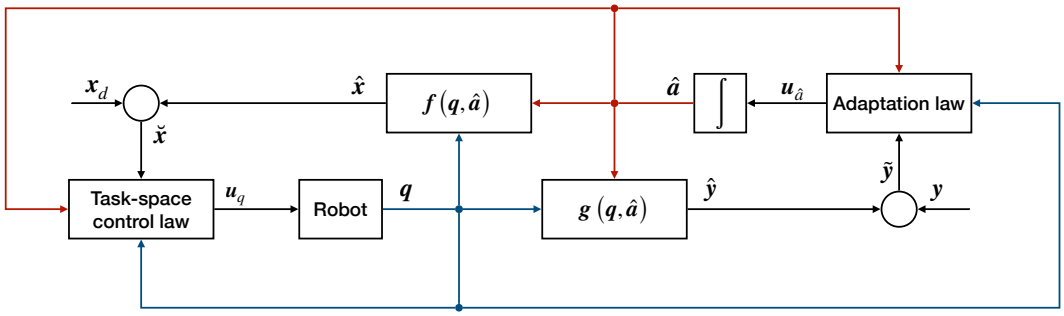


Fig. 1. Block diagram of the adaptive constrained controller. The *task-space control law* (6) aims to reduce the estimated task-space error $\check{\boldsymbol{x}}$ using the current robot configuration vector $\boldsymbol{q}$, the estimated parameter vector $\hat{\boldsymbol{a}}$, and the current desired task-space value $\boldsymbol{x}_d$ to compute the next optimal control input $\boldsymbol{u}_q$ sent to the robot. The *adaptation law* (12) aims to reduce the measurement error $\tilde{\boldsymbol{y}}$ using $\boldsymbol{q}$, $\hat{\boldsymbol{a}}$, and the current task-space measurement $\boldsymbol{y}$ to compute the optimal adaptation signal $\boldsymbol{u}_{\hat{a}}$. The red lines indicate the feedback of the parameter vector $\hat{\boldsymbol{a}}$ whereas the blue lines indicate the feedback of the robot configuration vector $\boldsymbol{q}$.

### A. Task-space control law

The control input is obtained as

$$\begin{aligned} \boldsymbol{u}_q \in \underset{\dot{\boldsymbol{q}}}{\operatorname{argmin}} \quad & \|\boldsymbol{J}_{\hat{x},q}\dot{\boldsymbol{q}} + \eta_q \check{\boldsymbol{x}}\|_2^2 + \|\boldsymbol{\Lambda}_q \dot{\boldsymbol{q}}\|_2^2 \\ \text{subject to} \quad & \boldsymbol{B}_q \dot{\boldsymbol{q}} \preceq \boldsymbol{b}_q \\ & \boldsymbol{W}_q \dot{\boldsymbol{q}} \preceq \boldsymbol{w}_q, \end{aligned} \tag{6}$$

where $\eta_q \in (0, \infty)$ is a proportional gain and $\boldsymbol{\Lambda}_q \in \mathbb{R}^{n\times n}$ is a positive definite gain matrix, usually diagonal, that penalizes high joint-velocities. In addition, $\boldsymbol{J}_{\hat{x},q}$ is the estimated task Jacobian, as defined in (3), and $\check{\boldsymbol{x}} \triangleq \check{\boldsymbol{x}}(\hat{\boldsymbol{x}}(t), \boldsymbol{x}_d)$ is a suitable *task error* that quantifies how close the estimated task-space variable is to the desired setpoint. A common choice is $\check{\boldsymbol{x}} \triangleq \hat{\boldsymbol{x}} - \boldsymbol{x}_d$. Furthermore, since $\hat{\boldsymbol{x}}$ depends on $\hat{\boldsymbol{a}}$, as shown in (2), the objective function in (6) also depends on the estimated parameters $\hat{\boldsymbol{a}}$.

If $\check{\boldsymbol{x}} = \boldsymbol{0}$, then $\boldsymbol{u}_q = \boldsymbol{0}$. This is due to the fact that when $\check{\boldsymbol{x}} = \boldsymbol{0}$, the objective function in (6) boils down to $\|\boldsymbol{J}_{\hat{x},q}\dot{\boldsymbol{q}}\|_2^2 + \|\boldsymbol{\Lambda}_q\dot{\boldsymbol{q}}\|_2^2$, which is minimized if and only if $\dot{\boldsymbol{q}} = \boldsymbol{0}$. This happens even when $\boldsymbol{J}_{\hat{x},q}$ has a non trivial nullspace because $\boldsymbol{\Lambda}_q$ is positive definite, hence invertible.

*Constraints:* The matrix $\boldsymbol{B}_q \triangleq \boldsymbol{B}_q(\boldsymbol{q}, \hat{\boldsymbol{a}}) \in \mathbb{R}^{s\times n}$ and $\boldsymbol{b} \triangleq \boldsymbol{b}(\boldsymbol{q}, \hat{\boldsymbol{a}}) \in \mathbb{R}^s$, with $\boldsymbol{b}_q \succeq \boldsymbol{0}$, define the $s$ linear (scalar) constraints imposed on the control inputs that also depend on the estimated parameters. Those constraints can be used to prevent self-collisions or collisions with other objects in the workspace using VFIs [12], as explained in Section III-C.

The constraint $\boldsymbol{W}_q\dot{\boldsymbol{q}} \preceq \boldsymbol{w}_q$, where $\boldsymbol{W}_q \triangleq \boldsymbol{W}_q(\boldsymbol{q}) \in \mathbb{R}^{s_q\times n}$ and $\boldsymbol{w} \triangleq \boldsymbol{w}(\boldsymbol{q}) \in \mathbb{R}^{s_q}$, is used to enforce the $s_q$ (scalar) constraints unrelated to the estimated parameters. For instance, configuration velocity limits $\boldsymbol{q}_{v,\min}$, $\boldsymbol{q}_{v,\max}$ can be trivially enforced with

$$\begin{bmatrix} -\boldsymbol{I}_{n\times n} \\ \boldsymbol{I}_{n\times n} \end{bmatrix} \dot{\boldsymbol{q}} \preceq \begin{bmatrix} -\boldsymbol{q}_{v,\min} \\ \boldsymbol{q}_{v,\max} \end{bmatrix}. \tag{7}$$

Moreover, to define limits for the robot configurations, we start by defining $\tilde{\boldsymbol{q}}_{\max}(t) \triangleq \boldsymbol{q}(t) - \boldsymbol{q}_{\max}$ and $\tilde{\boldsymbol{q}}_{\min}(t) \triangleq \boldsymbol{q}(t) - \boldsymbol{q}_{\min}$, such that $\tilde{\boldsymbol{q}}_{\max}(0) \preceq \boldsymbol{0}$ and $\tilde{\boldsymbol{q}}_{\min}(0) \succeq \boldsymbol{0}$, with the corresponding differential inequalities (notice that the joint limits do not vary with time, i.e., $\dot{\boldsymbol{q}}_{\min} = \dot{\boldsymbol{q}}_{\max} = \boldsymbol{0}$)

$$\dot{\tilde{\boldsymbol{q}}}_{\min}(t) + \eta_{\tilde{q}}\tilde{\boldsymbol{q}}_{\min}(t) \succeq \boldsymbol{0}, \tag{8}$$

$$\dot{\tilde{\boldsymbol{q}}}_{\max}(t) + \eta_{\tilde{q}}\tilde{\boldsymbol{q}}_{\max}(t) \preceq \boldsymbol{0}. \tag{9}$$

By Gronwall's Lemma [38], (9) and (8) ensure that $\tilde{\boldsymbol{q}}_{\max}(t) \preceq e^{-\eta_{\tilde{q}}t}\tilde{\boldsymbol{q}}_{\max}(0) \preceq \boldsymbol{0}$ and $\tilde{\boldsymbol{q}}_{\min}(t) \succeq e^{-\eta_{\tilde{q}}t}\tilde{\boldsymbol{q}}_{\min}(0) \succeq \boldsymbol{0}$ and can be rewritten as

$$\begin{bmatrix} -\boldsymbol{I}_{n\times n} \\ \boldsymbol{I}_{n\times n} \end{bmatrix} \dot{\boldsymbol{q}} \preceq -\eta_{\tilde{q}} \begin{bmatrix} -\tilde{\boldsymbol{q}}_{\min}(t) \\ \tilde{\boldsymbol{q}}_{\max}(t) \end{bmatrix}. \tag{10}$$

The configuration-only constraints can therefore be the composition of (7) and (10)

$$\underbrace{\begin{bmatrix} -\boldsymbol{I}_{n\times n} \\ \boldsymbol{I}_{n\times n} \\ -\boldsymbol{I}_{n\times n} \\ \boldsymbol{I}_{n\times n} \end{bmatrix}}_{\boldsymbol{W}_q} \dot{\boldsymbol{q}} \preceq \underbrace{\begin{bmatrix} -\boldsymbol{q}_{v,\min} \\ \boldsymbol{q}_{v,\max} \\ \eta_{\tilde{q}}\tilde{\boldsymbol{q}}_{\min}(t) \\ -\eta_{\tilde{q}}\tilde{\boldsymbol{q}}_{\max}(t) \end{bmatrix}}_{\boldsymbol{w}_q}. \tag{11}$$

### B. Adaptation law

The adaptation law is given by

$$\begin{aligned} \boldsymbol{u}_{\hat{a}} \in \underset{\dot{\hat{\boldsymbol{a}}}}{\operatorname{argmin}} \quad & \left\|\boldsymbol{J}_{\hat{y},\hat{a}}\dot{\hat{\boldsymbol{a}}} + \eta_{\hat{a}}\tilde{\boldsymbol{y}}\right\|_2^2 + \left\|\boldsymbol{\Lambda}_{\hat{a}}\dot{\hat{\boldsymbol{a}}}\right\|_2^2 \\ \text{subject to} \quad & \boldsymbol{B}_{\hat{a}}\dot{\hat{\boldsymbol{a}}} \preceq \boldsymbol{b}_{\hat{a}} \\ & \boldsymbol{W}_{\hat{a}}\dot{\hat{\boldsymbol{a}}} \preceq \boldsymbol{w}_{\hat{a}} \\ & \boldsymbol{N}_{\hat{a}}\dot{\hat{\boldsymbol{a}}} = \boldsymbol{0} \\ & \check{\boldsymbol{x}}^T\boldsymbol{J}_{\hat{x},\hat{a}}\dot{\hat{\boldsymbol{a}}} \leq 0, \end{aligned} \tag{12}$$

where $\eta_{\hat{a}} \in (0, \infty)$ is a proportional gain and $\boldsymbol{\Lambda}_{\hat{a}} \in \mathbb{R}^{p\times p}$ is a positive definite gain matrix, usually diagonal. In addition, $\tilde{\boldsymbol{y}} \triangleq \tilde{\boldsymbol{y}}(\hat{\boldsymbol{y}}(t), \boldsymbol{y}(t))$ is the *estimation error*, that is, the error between the estimated measure-space value and the real measure-space value. The estimation error indirectly measures the parameter estimation error using the measure-space estimated FKM. For example, $\tilde{\boldsymbol{y}} \triangleq \hat{\boldsymbol{y}} - \boldsymbol{y}$. Nonetheless, any differentiable error function that satisfies

$$\tilde{\boldsymbol{y}} = \boldsymbol{0} \iff \hat{\boldsymbol{y}} = \boldsymbol{y} \tag{13}$$

is acceptable.

The parameters can be updated even when the task error is zero, as long as the measurement error is not zero and the estimated task error is not affected. This can be interpreted as moving the end effector of a virtual robot made of estimated parameters $\hat{\boldsymbol{a}}$ in the direction of the real end-effector of the real robot made of actual parameters $\boldsymbol{a}$.[5] The term $\left\|\boldsymbol{\Lambda}_{\hat{a}}\dot{\hat{\boldsymbol{a}}}\right\|_2^2$ can be used to scale parameters that use different units, to minimize the detrimental effects that occur when the Jacobian is ill-conditioned, and to guarantee that the parameters stop being updated when $\hat{\boldsymbol{y}} = \boldsymbol{y} \iff \tilde{\boldsymbol{y}} = \boldsymbol{0}$. More specifically, $\tilde{\boldsymbol{y}} = \boldsymbol{0}$ implies $\left\|\boldsymbol{J}_{\hat{y},\hat{a}}\dot{\hat{\boldsymbol{a}}}\right\|_2^2 + \left\|\boldsymbol{\Lambda}_{\hat{a}}\dot{\hat{\boldsymbol{a}}}\right\|_2^2$, where $\boldsymbol{\Lambda}_{\hat{a}}$ is positive definite, which is minimized if and only if $\dot{\hat{\boldsymbol{a}}} = \boldsymbol{0}$.

Given the definition of the measure-space FKM (4) and (13),

$$\tilde{\boldsymbol{y}} = \boldsymbol{0} \iff \hat{\boldsymbol{y}} = \boldsymbol{y} \iff \boldsymbol{g}\left(\boldsymbol{q}, \hat{\boldsymbol{a}}\right) = \boldsymbol{g}\left(\boldsymbol{q}, \boldsymbol{a}\right).$$

However, as the measure-space FKM is in general not injective, we cannot say that $\boldsymbol{a} = \hat{\boldsymbol{a}}$ when $\tilde{\boldsymbol{y}} = \boldsymbol{0}$.

The adaptation law has four linear (vector) constraints. The first constraint is used to enforce task-space constraints that depend on the parameters, in which $\boldsymbol{B}_{\hat{a}} \triangleq \boldsymbol{B}_{\hat{a}}(\boldsymbol{q}, \hat{\boldsymbol{a}}) \in \mathbb{R}^{s\times p}$ and $\boldsymbol{b}_{\hat{a}} \triangleq \boldsymbol{b}_{\hat{a}}(\boldsymbol{q}, \hat{\boldsymbol{a}}) \in \mathbb{R}^{s}$, with $\boldsymbol{b}_{\hat{a}} \succeq \boldsymbol{0}$, and $s$ is the number of VFIs. That constraint is used to ensure that the first (vector) constraint in (6) is satisfied even during the adaptation of parameters, and explained in detail in Section III-C. The second constraint is used to enforce parameter bounds that are independent of the robot configuration, analogously to (8)–(10), with $\boldsymbol{W}_{\hat{a}} \in \mathbb{R}^{s_{\hat{a}}\times p}$ and $\boldsymbol{w}_{\hat{a}} \triangleq \boldsymbol{w}_{\hat{a}}\left(\hat{\boldsymbol{a}}\right) \in \mathbb{R}^{s_{\hat{a}}}$. The third constraint restricts the trajectories of the estimated parameters to lie within an invariant set. This is to prevent disturbing unmeasured variables, with $\boldsymbol{N}_{\hat{a}} \triangleq \boldsymbol{N}_{\hat{a}}\left(\boldsymbol{q}, \hat{\boldsymbol{a}}\right) \in \mathbb{R}^{r\times p}$ being the parametric task-Jacobian projector. The fourth constraint is a Lyapunov constraint [39], explained in detail in Section III-D.

### C. Vector field inequalities in an adaptive framework

In both (6) and (12), we have constraints that are used to enforce VFIs [12]; namely, $\boldsymbol{B}_q\dot{\boldsymbol{q}} \preceq \boldsymbol{b}_q$ and $\boldsymbol{B}_{\hat{a}}\dot{\hat{\boldsymbol{a}}} \preceq \boldsymbol{b}_{\hat{a}}$. The main idea of VFIs is as follows. Given $s$ differentiable functions $h_i : \mathcal{Q}\times\mathcal{A} \to \mathbb{R}$, such that $h_i \triangleq h_i(\boldsymbol{q}(t), \hat{\boldsymbol{a}}(t))$, we define the feasible set as

$$\mathscr{F} = \left\{ (\boldsymbol{q}, \hat{\boldsymbol{a}}) \in \mathcal{Q}\times\mathcal{A} : \bigwedge_{i=1}^{s} \left(h_i\left(\boldsymbol{q}, \hat{\boldsymbol{a}}\right) \leq 0\right) \right\}.$$

The $s$ VFIs are defined as $\dot{h}_i(t) + \eta_v h_i(t) \leq 0$, where $\eta_v \in (0, \infty)$. Let $\boldsymbol{h} = \begin{bmatrix} h_1 & \cdots & h_s \end{bmatrix}^T \in \mathbb{R}^s$, then $\dot{\boldsymbol{h}} = \begin{bmatrix} \boldsymbol{B}_q & \boldsymbol{B}_{\hat{a}} \end{bmatrix}\dot{\boldsymbol{v}}$, with $\boldsymbol{v} \triangleq \begin{bmatrix} \boldsymbol{q}^T & \hat{\boldsymbol{a}}^T \end{bmatrix}^T$,

$$\boldsymbol{B}_q = \begin{bmatrix} \nabla_q(h_1) & \cdots & \nabla_q(h_s) \end{bmatrix}^T,$$
$$\boldsymbol{B}_{\hat{a}} = \begin{bmatrix} \nabla_{\hat{a}}(h_1) & \cdots & \nabla_{\hat{a}}(h_s) \end{bmatrix}^T,$$

in which $\nabla_q(h_i) \triangleq \begin{bmatrix} \partial h_i/\partial q_1 & \cdots & \partial h_i/\partial q_n \end{bmatrix}^T$ and $\nabla_{\hat{a}}(h_i) \triangleq \begin{bmatrix} \partial h_i/\partial \hat{a}_1 & \cdots & \partial h_i/\partial \hat{a}_p \end{bmatrix}^T$.

From Gronwall's lemma, given the differential inequality $\dot{\boldsymbol{h}}(t) + \eta_v\boldsymbol{h}(t) \preceq \boldsymbol{0}$, if $\boldsymbol{h}(0) \preceq \boldsymbol{0}$ when $t = 0$, then $\boldsymbol{h}(t) \preceq e^{-\eta_v t}\boldsymbol{h}(0) \preceq \boldsymbol{0}$ for all $t \geq 0$. Thus, we stack the $s$ scalar VFIs to obtain a single differential inequality in vector form:

$$\boldsymbol{B}_q\dot{\boldsymbol{q}} + \boldsymbol{B}_{\hat{a}}\dot{\hat{\boldsymbol{a}}} \preceq -\eta_v\boldsymbol{h}(t). \tag{14}$$

To ensure (14) by means of the first inequality in (6) and the first inequality in (12), we define $\boldsymbol{b}_q \triangleq -\eta_v\boldsymbol{h}(t)\left(1-\alpha\right)$ and $\boldsymbol{b}_{\hat{a}} \triangleq -\eta_v\boldsymbol{h}(t)\alpha$, with $\alpha \in [0, 1]$, such that

$$\left(\boldsymbol{B}_q\dot{\boldsymbol{q}} \preceq \boldsymbol{b}_q\right) \wedge \left(\boldsymbol{B}_{\hat{a}}\dot{\hat{\boldsymbol{a}}} \preceq \boldsymbol{b}_{\hat{a}}\right). \tag{15}$$

### D. Closed-loop stability

To prove that the closed-loop system composed of (3), (5), (6), and (12) is stable, we first define *equivalent* optimization problems to (6) and (12). Then, we show that under Assumptions 2 and 3 of Section II-B, those optimization problems are always feasible, which means that it suffices to analyze the objective functions to determine closed-loop stability. Finally, we choose a Lyapunov function candidate and exploit the optimal solution to prove that the closed-loop system is stable.

*Equivalent optimization problems:* We use the extended vector $\boldsymbol{v} = \begin{bmatrix} \boldsymbol{q}^T & \hat{\boldsymbol{a}}^T \end{bmatrix}^T$ such that

$$\begin{aligned} \boldsymbol{u}_{v,q} \in \underset{\dot{\boldsymbol{v}}}{\operatorname{argmin}} \quad & \left\|\boldsymbol{J}_{\hat{x}}\dot{\boldsymbol{v}} + \eta_q\breve{\boldsymbol{x}}\right\|_2^2 + \left\|\boldsymbol{\Lambda}_{v,q}\dot{\boldsymbol{v}}\right\|_2^2 \\ \text{subject to} \quad & \boldsymbol{B}_{v,q}\dot{\boldsymbol{v}} \preceq \boldsymbol{b}_q \\ & \boldsymbol{W}_{v,q}\dot{\boldsymbol{v}} \preceq \boldsymbol{w}_q \\ & \boldsymbol{A}_q\dot{\boldsymbol{v}} = \boldsymbol{0}, \end{aligned} \tag{16}$$

in which $\boldsymbol{J}_{\hat{x}} = \begin{bmatrix} \boldsymbol{J}_{\hat{x},q} & \boldsymbol{J}_{\hat{x},\hat{a}} \end{bmatrix}$, $\boldsymbol{\Lambda}_{v,q} = \begin{bmatrix} \boldsymbol{\Lambda}_q & \boldsymbol{0}_{n\times p} \end{bmatrix} \in \mathbb{R}^{n\times(n+p)}$, $\boldsymbol{B}_{v,q} = \begin{bmatrix} \boldsymbol{B}_q & \boldsymbol{0}_{s\times p} \end{bmatrix} \in \mathbb{R}^{s\times(n+p)}$, $\boldsymbol{W}_{v,q} = \begin{bmatrix} \boldsymbol{W}_q & \boldsymbol{0}_{s_q\times p} \end{bmatrix} \in \mathbb{R}^{s_q\times(n+p)}$, and $\boldsymbol{A}_q = \operatorname{blkdiag}(\boldsymbol{0}_{n\times n}, \boldsymbol{I}_{p\times p}) \in \mathbb{R}^{(n+p)\times(n+p)}$. Notice that (16) boils down to (6). More specifically, $\boldsymbol{B}_{v,q}\dot{\boldsymbol{v}} = \boldsymbol{B}_q\dot{\boldsymbol{q}}$ and $\boldsymbol{W}_{v,q}\dot{\boldsymbol{v}} = \boldsymbol{W}_q\dot{\boldsymbol{q}}$. Also, $\boldsymbol{A}_q\dot{\boldsymbol{v}} = \boldsymbol{0}$ implies $\dot{\hat{\boldsymbol{a}}} = \boldsymbol{0}$. (It also implies $\boldsymbol{0}_{n\times n}\dot{\boldsymbol{q}} = \boldsymbol{0}$, which is always true regardless the value of $\dot{\boldsymbol{q}}$). Hence,

$$\boldsymbol{J}_{\hat{x}}\dot{\boldsymbol{v}} + \eta_q\breve{\boldsymbol{x}} = \boldsymbol{J}_{\hat{x},q}\dot{\boldsymbol{q}} + \boldsymbol{J}_{\hat{x},\hat{a}}\dot{\hat{\boldsymbol{a}}} + \eta_q\breve{\boldsymbol{x}} = \boldsymbol{J}_{\hat{x},q}\dot{\boldsymbol{q}} + \eta_q\breve{\boldsymbol{x}}.$$

Lastly, $\boldsymbol{\Lambda}_{v,q}\dot{\boldsymbol{v}} = \boldsymbol{\Lambda}_q\dot{\boldsymbol{q}}$ and $\boldsymbol{u}_{v,q} = \begin{bmatrix} \boldsymbol{u}_q^T & \boldsymbol{0}^T \end{bmatrix}^T$.

Analogously,

$$\begin{aligned} \boldsymbol{u}_{v,\hat{a}} \in \underset{\dot{\boldsymbol{v}}}{\operatorname{argmin}} \quad & \left\|\boldsymbol{J}_{\hat{y}}\dot{\boldsymbol{v}} + \eta_{\hat{a}}\tilde{\boldsymbol{y}}\right\|_2^2 + \left\|\boldsymbol{\Lambda}_{v,\hat{a}}\dot{\boldsymbol{v}}\right\|_2^2 \\ \text{subject to} \quad & \boldsymbol{B}_{v,\hat{a}}\dot{\boldsymbol{v}} \preceq \boldsymbol{b}_{\hat{a}} \\ & \boldsymbol{W}_{v,\hat{a}}\dot{\boldsymbol{v}} \preceq \boldsymbol{w}_{\hat{a}} \\ & \boldsymbol{N}_{v,\hat{a}}\dot{\boldsymbol{v}} = \boldsymbol{0} \\ & \breve{\boldsymbol{x}}^T\boldsymbol{J}_{\hat{x}}\dot{\boldsymbol{v}} \leq 0 \\ & \boldsymbol{A}_{\hat{a}}\dot{\boldsymbol{v}} = \boldsymbol{0}, \end{aligned} \tag{17}$$

[5] The attentive reader will notice that, when $\breve{\boldsymbol{x}} = \boldsymbol{0}$, the last constraint in (12) is innocuous because it becomes $0 \leq 0$, which is always true regardless of $\dot{\hat{\boldsymbol{a}}}$. However, in this situation, the trajectories of the closed-loop system have already converged to the invariant set determined by $\dot{V}(\breve{\boldsymbol{x}}) = 0$, as shown in Section III-D, which means that the norm of the task error cannot increase despite changes in the estimated parameters $\hat{\boldsymbol{a}}$.

in which $\boldsymbol{J}_{\hat{y}} = \begin{bmatrix}\boldsymbol{J}_{\hat{y},q} & \boldsymbol{J}_{\hat{y},\hat{a}}\end{bmatrix}$, $\boldsymbol{\Lambda}_{v,\hat{a}} = \begin{bmatrix}\boldsymbol{0}_{p\times n} & \boldsymbol{\Lambda}_{\hat{a}}\end{bmatrix} \in \mathbb{R}^{p\times(n+p)}$, $\boldsymbol{B}_{v,\hat{a}} = \begin{bmatrix}\boldsymbol{0}_{s\times n} & \boldsymbol{B}_{\hat{a}}\end{bmatrix} \in \mathbb{R}^{s\times(n+p)}$, $\boldsymbol{W}_{v,\hat{a}} = \begin{bmatrix}\boldsymbol{0}_{s_{\hat{a}}\times n} & \boldsymbol{W}_{\hat{a}}\end{bmatrix} \in \mathbb{R}^{s_{\hat{a}}\times(n+p)}$, $\boldsymbol{N}_{v,\hat{a}} = \begin{bmatrix}\boldsymbol{0}_{\bar{r}\times n} & \boldsymbol{N}_{\hat{a}}\end{bmatrix} \in \mathbb{R}^{\bar{r}\times(n+p)}$, and $\boldsymbol{A}_{\hat{a}} = \text{blkdiag}(\boldsymbol{I}_{n\times n}, \boldsymbol{0}_{p\times p}) \in \mathbb{R}^{(n+p)\times(n+p)}$. Again, it is easy to show that (17) is equivalent to (12). First, by direct calculation we find that the first three constraints in (17) boil down to the first three constraints in (12). Also, the constraint $\boldsymbol{A}_{\hat{a}}\dot{\boldsymbol{v}} = \boldsymbol{0}$ ensures that $\dot{\boldsymbol{q}} = \boldsymbol{0}$. Therefore, $\breve{\boldsymbol{x}}^T\boldsymbol{J}_{\hat{x}}\dot{\boldsymbol{v}} = \breve{\boldsymbol{x}}^T\boldsymbol{J}_{\hat{x},\hat{a}}\dot{\hat{\boldsymbol{a}}}$. Lastly, because $\dot{\boldsymbol{q}} = \boldsymbol{0}$, the objective function in (17) reduces to the one in (12), and $\boldsymbol{u}_{v,\hat{a}} = \begin{bmatrix}\boldsymbol{0}^T & \boldsymbol{u}_{\hat{a}}^T\end{bmatrix}^T$.

*Feasibility of the optimization problems:* Assumption 2 in Section II-B determines that $\boldsymbol{v}(0) \in \mathscr{F}$, which implies that $\boldsymbol{h}(0) \preceq \boldsymbol{0}$ and, consequently, $\boldsymbol{b}_q(0), \boldsymbol{b}_{\hat{a}}(0) \succeq \boldsymbol{0}$. Because $\dot{\boldsymbol{v}} = \boldsymbol{0}$ is a feasible solution by Assumption 3, the optimization problems (16) and (17) are always feasible. In other words, in the worst case, the robot can stop. Hence, it is sufficient to analyze only the objective function to determine closed-loop stability.

*Remark* 1. Gronwall's lemma [38] guarantees that the system composed of (3), (5), (6), and (12) will never violate a constraint in the estimated task-space. However, this does not mean that it is guaranteed that there will be no violation in the real task-space. A practical solution to this problem is shown in Section VI, which consists of giving the system some time to adjust the estimated model to be closer to the real one before the robot starts moving.

*Lyapunov stability:* Choosing the Lyapunov candidate $V(\breve{\boldsymbol{x}}) \triangleq V = (1/2)\,\breve{\boldsymbol{x}}^T\breve{\boldsymbol{x}}$, in which $\breve{\boldsymbol{x}} = \hat{\boldsymbol{x}} - \boldsymbol{x}_d$, we obtain $\dot{V} = \breve{\boldsymbol{x}}^T\dot{\hat{\boldsymbol{x}}} = \breve{\boldsymbol{x}}^T\boldsymbol{J}_{\hat{x}}\dot{\boldsymbol{v}}$. Since $\boldsymbol{u}_{v,q} = \begin{bmatrix}\boldsymbol{u}_q^T & \boldsymbol{0}^T\end{bmatrix}^T$ and $\boldsymbol{u}_{v,\hat{a}} = \begin{bmatrix}\boldsymbol{0}^T & \boldsymbol{u}_{\hat{a}}^T\end{bmatrix}^T$, then $\dot{\boldsymbol{v}} = \boldsymbol{u}_{v,q} + \boldsymbol{u}_{v,\hat{a}}$. Therefore,

$$\dot{V} = \breve{\boldsymbol{x}}^T\boldsymbol{J}_{\hat{x}}\dot{\boldsymbol{v}} = \breve{\boldsymbol{x}}^T\boldsymbol{J}_{\hat{x}}\boldsymbol{u}_{v,q} + \breve{\boldsymbol{x}}^T\boldsymbol{J}_{\hat{x}}\boldsymbol{u}_{v,\hat{a}}. \tag{18}$$

Now we show that each term of the right-hand side of (18) is non-positive. From (16), we have

$$\|\boldsymbol{J}_{\hat{x}}\boldsymbol{u}_{v,q} + \eta_q\breve{\boldsymbol{x}}\|_2^2 + \|\boldsymbol{\Lambda}_{v,q}\boldsymbol{u}_{v,q}\|_2^2 \leq \|\eta_q\breve{\boldsymbol{x}}\|_2^2$$

because $\boldsymbol{u}_{v,q}$ is the optimal solution and $\dot{\boldsymbol{v}} = \boldsymbol{0}$ is always a feasible solution by assumption. Therefore,

$$\|\boldsymbol{J}_{\hat{x}}\boldsymbol{u}_{v,q}\|_2^2 + 2\eta_q\breve{\boldsymbol{x}}^T\boldsymbol{J}_{\hat{x}}\boldsymbol{u}_{v,q} + \|\eta_q\breve{\boldsymbol{x}}\|_2^2 + \|\boldsymbol{\Lambda}_{v,q}\boldsymbol{u}_{v,q}\|_2^2 \leq \|\eta_q\breve{\boldsymbol{x}}\|_2^2,$$

which implies

$$\breve{\boldsymbol{x}}^T\boldsymbol{J}_{\hat{x}}\boldsymbol{u}_{v,q} \leq -\frac{1}{2\eta_q}\left(\|\boldsymbol{J}_{\hat{x}}\boldsymbol{u}_{v,q}\|_2^2 + \|\boldsymbol{\Lambda}_{v,q}\boldsymbol{u}_{v,q}\|_2^2\right) \leq 0. \tag{19}$$

From (17) we have that $\breve{\boldsymbol{x}}^T\boldsymbol{J}_{\hat{x}}\boldsymbol{u}_{v,\hat{a}} \leq 0$ is enforced by the optimization problem. Hence, (18) is nonpositive, which implies $\dot{V} = \breve{\boldsymbol{x}}^T\boldsymbol{J}_{\hat{x}}\dot{\boldsymbol{v}} \leq 0$. Thus, we conclude that the closed-loop system is stable.

*Remark* 2. Due to the generality of the constraints being considered, only closed-loop stability can be ensured. One intuitive way of illustrating this is to think of a manipulator robot holding an object trying to place it on a table nearby. If there is a wall between the robot and the table, the constraint will prevent the robot from reaching the table, making it physically impossible to have asymptotic convergence without piercing through the wall. Notice that this is the desired behavior: the safety of the robot and its environment are prioritized over asymptotic convergence.

*Remark* 3. When using non-additive errors (e.g., when using unit quaternions to represent orientations or unit dual quaternions to represent poses), it is possible to use suitable projectors such that the argument above still holds. See Appendix III.

## IV. Use cases

TABLE II

Parametric measure-space estimated Jacobian and parametric task-Jacobian projector.

| | $\mathcal{Y}_x$ | $\mathcal{Y}_r$ | $\mathcal{Y}_t$ | $\mathcal{Y}_d$ |
|---|---|---|---|---|
| $\boldsymbol{J}_{\hat{y},\hat{a}}$ | $\boldsymbol{J}_{x,\hat{a}}$ | $\boldsymbol{J}_{r,\hat{a}}$ | $\boldsymbol{J}_{t,\hat{a}}$ | $\boldsymbol{J}_{d,\hat{a}}$ |
| $\boldsymbol{N}_{\hat{a}}$ | $\boldsymbol{0}$ | $\boldsymbol{J}_{t,\hat{a}}$ | $\boldsymbol{J}_{r,\hat{a}}$ | (20) |

Notice that $\boldsymbol{J}_{x,\hat{a}} \in \mathbb{R}^{m_x\times n}$, $\boldsymbol{J}_{r,\hat{a}} \in \mathbb{R}^{m_r\times n}$, $\boldsymbol{J}_{t,\hat{a}} \in \mathbb{R}^{3\times n}$, and $\boldsymbol{J}_{d,\hat{a}} \in \mathbb{R}^{1\times n}$ are the parametric Jacobians that satisfy (3) for the estimated pose, rotation, translation, and distance, respectively. The dimensions $m_x$ and $m_r$ depend on the parameterization used for the pose and rotation.

We summarize four relevant use-cases for the measurement space $\mathcal{Y}$ (i.e., the space of relevant end-effector measurements with respect to the sensor's reference frame $\mathcal{F}_w$), the estimated measure-space Jacobian $\boldsymbol{J}_{\hat{y},\hat{a}}$, and the parametric task-Jacobian projector $\boldsymbol{N}_{\hat{a}}$ in Table II. A more detailed description of how each measurement space is parameterized is given in Appendix II.

*Measurement of complete end-effector pose:* The measurement space is the end-effector pose space $\mathcal{Y}_x$ (Appendix II-C). Given that this represents a complete measurement, the parametric task-Jacobian projector is trivial, i.e. $\boldsymbol{N}_{\hat{a}} \triangleq \boldsymbol{0}$.

*Measurement of end-effector orientation:* The measurement space is the end-effector rotation space $\mathcal{Y}_r$ (Appendix II-B). In this case, the real translation $\boldsymbol{t}$ is unknown, so we enforce $\dot{\hat{\boldsymbol{t}}} = \boldsymbol{0}$, which is achieved when $\boldsymbol{N}_{\hat{a}} \triangleq \boldsymbol{J}_{t,\hat{a}}$.

*Measurement of end-effector translation:* The measurement space is the end-effector translation space $\mathcal{Y}_t$ (Appendix II-A). In this case, the real rotation $\boldsymbol{r}$ is unknown; hence, we constrain any parameter update on the rotation by enforcing $\dot{\hat{\boldsymbol{r}}} = \boldsymbol{0}$, which is achieved when $\boldsymbol{N}_{\hat{a}} \triangleq \boldsymbol{J}_{r,\hat{a}}$.

*Measurement of the end-effector distance:* The measurement space consists of the (Euclidean) distance space, $\mathcal{Y}_d \triangleq \left\{\|\boldsymbol{p}\|_2 : \boldsymbol{p} \in \mathbb{R}^3\right\}$, which contains all possible distances between the end-effector position and the inertial reference-frame. When we measure only the Euclidean distance of the end-effector to the origin of the reference frame, we constrain any rotation estimation update. In addition, we constrain the position estimation updates to the line that connects the current end-effector position to the origin of the reference frame, as shown in Fig. 2. This is because moving along that line ensures that the estimated end-effector distance improves while the position estimation does not worsen, as shown in the following lemma.

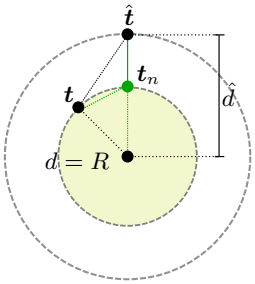


Fig. 2. All distances are measured with respect to the origin of the sphere, and $\boldsymbol{t}, \hat{\boldsymbol{t}} \in \mathbb{R}^3$ are the real and estimated positions, respectively. The figure shows the circle formed by the intersection of the sphere and the plane containing $\boldsymbol{t}$, $\hat{\boldsymbol{t}}$, and the sphere's center. The solid green line represents all $\boldsymbol{t}_\lambda \in \mathbb{R}^3$ along the line that connects $\hat{\boldsymbol{t}}$ to the center, such that $R \leq \|\boldsymbol{t}_\lambda\|_2 \leq \left\|\hat{\boldsymbol{t}}\right\|_2$ and $\hat{\boldsymbol{t}} \times \boldsymbol{t}_\lambda = \boldsymbol{0}$. Distance estimation updates in that line guarantee that the distance estimation improves and the position estimation given by $\left\|\hat{\boldsymbol{t}} - \boldsymbol{t}\right\|_2$ does not worsen. The green point $\boldsymbol{t}_n$ satisfies $\|\boldsymbol{t}_n\| = R$ with $\hat{\boldsymbol{t}} \times \boldsymbol{t}_n = \boldsymbol{0}$.

TABLE III
DH PARAMETERS OF THE VS050 ROBOT.

| | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|
| $\theta_{\text{DH}}$ [rad] | $-\pi$ | $\pi/2$ | $-\pi/2$ | 0 | $\pi$ | 0 |
| $d_{\text{DH}}$ [m] | 0.345 | 0 | 0 | 0.255 | 0 | 0.07 |
| $a_{\text{DH}}$ [m] | 0 | 0.250 | 0.01 | 0 | 0 | 0 |
| $\alpha_{\text{DH}}$ [rad] | $\pi/2$ | 0 | $-\pi/2$ | $\pi/2$ | $\pi/2$ | 0 |

**Lemma 4.** *Consider a sphere of radius $R$, an arbitrary point $\boldsymbol{t}$ on the sphere's surface, and a point $\hat{\boldsymbol{t}}$ outside of it. Also, let $\boldsymbol{t}_n = R\hat{\boldsymbol{t}}/\left\|\hat{\boldsymbol{t}}\right\|$ and $\lambda \in [0, 1]$. Assuming the center of the sphere as the reference point, the distance between any point $\boldsymbol{t}_\lambda = (1-\lambda)\hat{\boldsymbol{t}} + \lambda\boldsymbol{t}_n$ and $\boldsymbol{t}$ is smaller or equal than the distance between $\hat{\boldsymbol{t}}$ and $\boldsymbol{t}$.*

See Appendix I. Therefore, when preventing the update of the rotation estimation and constraining the position estimation update to the line that connects the current end-effector position to the origin of the reference frame, we obtain

$$\begin{cases} \dot{\hat{\boldsymbol{r}}} = \boldsymbol{0} \\ \hat{\boldsymbol{t}} \times \dot{\hat{\boldsymbol{t}}} = \boldsymbol{0} \end{cases} \implies \underbrace{\begin{bmatrix} \boldsymbol{J}_{r,\hat{a}} \\ \boldsymbol{S}\left(\hat{\boldsymbol{t}}\right)\boldsymbol{J}_{t,\hat{a}} \end{bmatrix}}_{\boldsymbol{N}_{d,\hat{a}}} \dot{\hat{\boldsymbol{a}}} = \boldsymbol{0}, \tag{20}$$

in which $\boldsymbol{S}(\cdot) : \mathbb{R}^3 \to \text{so}(3)$ is a skew-symmetric matrix such that $\boldsymbol{S}(\boldsymbol{a})\boldsymbol{b} = \boldsymbol{a} \times \boldsymbol{b}$, with $\boldsymbol{a}, \boldsymbol{b} \in \mathbb{R}^3$, and $\boldsymbol{t} \in \mathbb{R}^3$ is the end-effector position with respect to $\mathcal{F}_w$.

## V. EXPERIMENT PM: PARTIAL MEASUREMENTS

The experimental setup[6] is composed of a six-DoF manipulator robot (VS050, DENSO WAVE, Japan) with coordinate system $\mathcal{F}_{\text{base}}$ and an image-based position sensor (Polaris Vega, NDI, Canada) with coordinate system $\mathcal{F}_{\text{sensor}}$. The image-based sensor was used to obtain the pose of the *tip marker* with respect to $\mathcal{F}_{\text{sensor}}$. The tip marker was attached to the tip of the robotic manipulator and is represented by $\mathcal{F}_{\text{tip}}$. These elements are shown in Fig. 3.

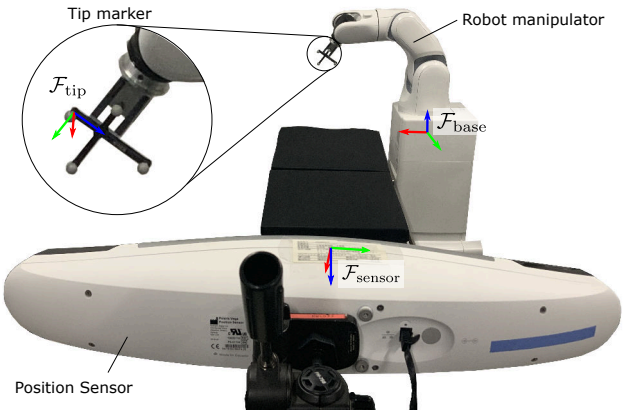


Fig. 3. Visualization of the experimental setup.

In this first experiment, the objective is to reach four consecutive setpoints in the robot's real task-space, which consists of the space of end-effector poses (i.e., position and orientation), while adapting the parameters of the robot. There is no collision avoidance because the purpose is to evaluate the closed-loop system behavior under uncertain parameters and complete or partial measurements. The parameters of the robot consist of all the robot's DH parameters and an initial rough estimate of $\mathcal{F}_{\text{base}}$ and $\mathcal{F}_{\text{tip}}$.

We used the manufacturer's documentation,[7] summarized in Table III, to obtain the initial value for the DH parameters.[8] The control law adapts the parameters $\theta_{\text{DH}}, d_{\text{DH}}, a_{\text{DH}}$, and $\alpha_{\text{DH}}$ of each joint, resulting in 24 joint-related parameters. The limits for the estimated translational parameters are of $\pm 1$ mm, and $\pm 1^\circ$ for the angular parameters.

The transformations $\mathcal{F}_{\text{base}}$ and $\mathcal{F}_{\text{tip}}$ are modeled using six parameters each. Those parameters describe six sequential transformations, namely a translation along the $x$-axis, a translation along the $y$-axis, a translation along the $z$-axis, a rotation about the $x$-axis, a rotation about the $y$-axis, and a rotation about the $z$-axis. These two transformations were roughly measured using a tape measure. The limits for the estimate of the translation parameters are $\pm 10$ cm from the initially measured values, and the limits for the estimate of the angular parameters are $\pm 20^\circ$ from the initially measured values. With these 12 parameters in conjunction with the joint-related parameters, a total of 36 parameters were estimated online (i.e., $p = 36$).

The experiments were executed under five conditions. Without any adaptation (**PM0**), and with adaptation using four subsets of the pose measurement: (**PM1**) pose, (**PM2**) rotation, (**PM3**) translation, and (**PM4**) distance. Although in **PM2–4** only partial measurements are used in the adaptation

[6]Software implementation based on `https://github.com/SmartArmStack` and DQ Robotics C++11 [40].

[7]`https://www.denso-wave.com/en/robot/product/five-six/vs050-060.html`

[8]We chose the DH parameters because they are ubiquitous in the literature on robot manipulators, but our methodology does not depend on specific parameterization.

law, we store the information of the complete end-effector pose measurements and use them as ground-truth in our analyses.

The control loop runs at 50 Hz ($T = 20\,\mathrm{ms}$), which is the maximum sampling frequency of our measurement system. The control gains are $\eta_q = \eta_{\hat{a}} = 40$, and the damping factors are $\boldsymbol{\Lambda}_q = 0.01\boldsymbol{I}_6$ and $\boldsymbol{\Lambda}_{\hat{a}} = 0.01\boldsymbol{I}_{36}$. The joint velocity limits were set at $\pm 0.2$ rad/s to partially compensate for the relatively low sampling time.

*Results and discussion*

The results regarding **PM0** and **PM1** are summarized in Fig. 4 in terms of real task error $\tilde{\boldsymbol{x}}$ (i.e., the error between the current measurement $\boldsymbol{y}$ and the desired pose $\boldsymbol{x}_d$), the measurement error $\tilde{\boldsymbol{y}}$ (i.e., the error between the current estimated pose $\hat{\boldsymbol{x}}$ and the current measurement $\boldsymbol{y}$), and the estimated task error $\breve{\boldsymbol{x}}$ (i.e., the error between the current estimated pose $\hat{\boldsymbol{x}}$ and the desired pose $\boldsymbol{x}_d$). Notice that when the pose error norm equals zero, the translation, rotation, and error norms also equal zero.

Regarding partial measurements, Fig. 5 shows the results of **PM0**, **PM2**, **PM3**, and **PM4** in terms of translation, rotation, and distance errors. Notice that when the translation error norm equals zero, the distance error norm also equals zero. The converse, however, is not true because the distance error can be zero while the translation error is not zero.

In our experiments, a given measurement space always caused the convergence of the real task error and measurement error in that specific dimension when the joint limits were not reached. For example, when the rotation is measured, the real rotation error norm goes asymptotically to zero, as shown by the solid-blue starred curve of the third graph in Fig. 5, and the measurement rotation error norm also goes to zero, as shown by the solid-blue starred curve of the forth graph in Fig. 5.

*No adaptation:* **PM0** represents the performance of a task controller without any parameter adaptation, which means that only (6) was used in the generation of control inputs. The real task error for all setpoints in Fig. 4 illustrates the errors in the initial estimation of the parameters. Because the robot parameters are all initially incorrect, no adaptation means that even if the estimated task error converges to zero, as shown by the solid-black curves in the third graph of Fig. 4 during the regulation to the second and fourth setpoints, there is no guarantee that the real-task error norm will converge to zero. The estimated task error could not converge for setpoints 1 and 3 because the robot reached its joint limits. This happened in spite of all setpoints being reachable in the real task-space. The measurement error when there is no adaptation can be understood as the results of a state-of-the-art constrained kinematic controller without adaptation [12], that we use as a comparative baseline for our adaptive strategies.

*Measurement of complete end-effector pose:* The complete pose measurement (**PM1**), represented by the solid red circled curves in Fig. 4, resulted in the convergence to zero of the real task error, the measurement error, and the estimated task error for all setpoints. Because of that, the translation, rotation, and error norms also converge to zero.

*Measurement of end-effector orientation:* When only rotation is measured (**PM2**), there was convergence of the estimated rotation error. It caused a reduction of the real pose error and the pose measurement error. The real and measurement translation errors remained similar to the unmeasured cases, thanks to the projection into the nullspace of the translation Jacobian, but the real and measurement distance errors worsened (fifth and sixth graphs in Fig. 5). Although this may seem counterintuitive, it is not unexpected. Indeed, let us consider the estimated position $\hat{\boldsymbol{p}}$ and real position $\boldsymbol{p}$ associated with the estimated end-effector pose $\hat{\boldsymbol{x}}$ and real end-effector pose $\boldsymbol{x}$, respectively. The translation measurement error is given by $\tilde{\boldsymbol{p}} = \hat{\boldsymbol{p}} - \boldsymbol{p}$ and the distance measurement error is given by $\tilde{d} = \|\hat{\boldsymbol{p}}\| - \|\boldsymbol{p}\|$. Hence, $\|\tilde{\boldsymbol{p}}\|^2 - \left\|\tilde{d}\right\|^2 = -2\hat{\boldsymbol{p}}^T\boldsymbol{p} + 2\,\|\hat{\boldsymbol{p}}\|\,\|\boldsymbol{p}\|$. Since $\hat{\boldsymbol{p}}^T\boldsymbol{p} = \|\hat{\boldsymbol{p}}\|\,\|\boldsymbol{p}\|\cos\phi$, where $\phi$ is the angle between $\hat{\boldsymbol{p}}$ and $\boldsymbol{p}$, we obtain $\left\|\tilde{d}\right\|^2 = \|\tilde{\boldsymbol{p}}\|^2 - 2\,\|\hat{\boldsymbol{p}}\|\,\|\boldsymbol{p}\|\,(1-\cos\phi)$. Now, consider another estimated position $\hat{\boldsymbol{p}}'$ and the associated translation measurement error $\tilde{\boldsymbol{p}}' = \hat{\boldsymbol{p}}' - \boldsymbol{p}$, such that $\|\tilde{\boldsymbol{p}}\| = \left\|\tilde{\boldsymbol{p}}'\right\|$. Analogously, let $\tilde{d}' = \left\|\hat{\boldsymbol{p}}'\right\| - \|\boldsymbol{p}\|$, then $\left\|\tilde{d}'\right\|^2 = \left\|\tilde{\boldsymbol{p}}'\right\|^2 - 2\left\|\hat{\boldsymbol{p}}'\right\|\|\boldsymbol{p}\|\,(1-\cos\phi')$, where $\phi'$ is the angle between $\hat{\boldsymbol{p}}'$ and $\boldsymbol{p}$. Thus,

$$\left\|\tilde{d}'\right\|^2 = \left\|\tilde{d}\right\|^2 + 2\,\|\boldsymbol{p}\|\left(\|\hat{\boldsymbol{p}}\|\,(1-\cos\phi) - \left\|\hat{\boldsymbol{p}}'\right\|\,(1-\cos\phi')\right)$$

because $\|\tilde{\boldsymbol{p}}\| = \left\|\tilde{\boldsymbol{p}}'\right\|$. If $\|\hat{\boldsymbol{p}}\|\,(1-\cos\phi) > \left\|\hat{\boldsymbol{p}}'\right\|\,(1-\cos\phi')$, then $\left\|\tilde{d}'\right\|^2 > \left\|\tilde{d}\right\|^2$, which implies $\left\|\tilde{d}'\right\| > \left\|\tilde{d}\right\|$.

*Measurement of end-effector translation:* When only the translation is measured (**PM3**), the real translation error and the measured translation error, indicated by the green squared curves in Fig. 5 converge to zero. Since zero translation error norm implies a zero distance error norm, the distance error also converged to zero. A better estimation of the translation allowed the estimated end-effector pose to converge to all setpoints without reaching the joint limits.

*Measurement of the end-effector distance:* In **PM4**, using the distance measurements only allowed for the convergence of measured distance, but not the real and estimated distances in setpoint 3 because the joint limits were reached, as illustrated by the pink crossed curves in Figs. 6 and 5.

These results correspond to the theoretical expectations. When the robot was unable to reach specific setpoints, it gracefully stopped moving in accordance with our proof of closed-loop stability. They also show the benefits of using partial information when only that information would be available. Depending on the requirements of a given task, the designer should choose an appropriate sensor by considering the trade-off between the expected improvement in real task error and the monetary/time costs related to using that sensor.

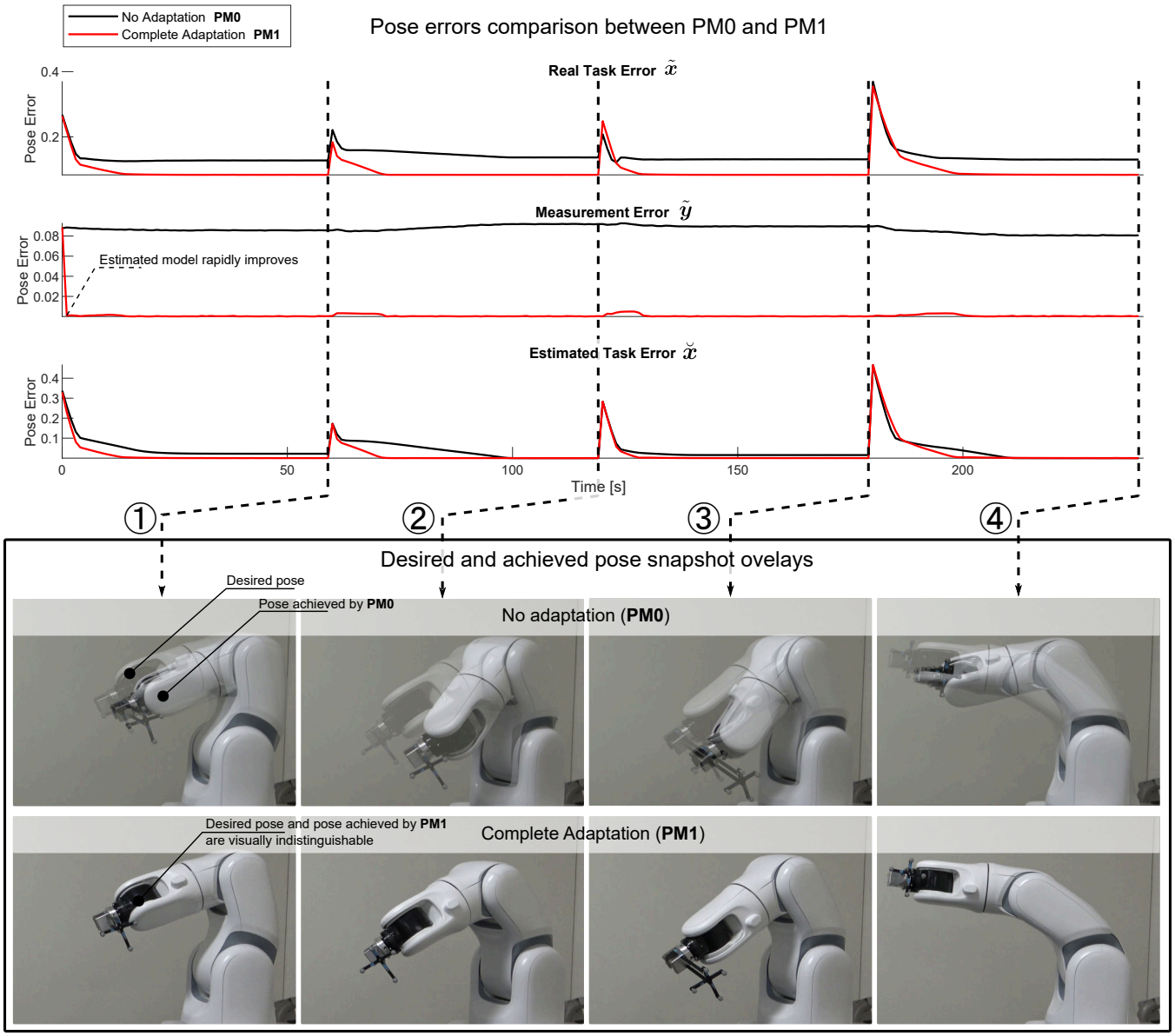


Fig. 4. Comparison of the pose error between **PM0** and **PM1**, in terms of real task pose error $\tilde{\boldsymbol{x}}$, estimated task pose error $\breve{\boldsymbol{x}}$, and measurement pose error $\tilde{\boldsymbol{y}}$. The proposed controller with complete measurements (**PM1**) outperforms a state-of-the-art kinematic controller [12] without adaptation (**PM0**) in all errors. The snapshots of the experiment show qualitatively the large final real task error in **PM0**, whereas the final real task error in **PM1** is indistinguishable from the real desired end-effector pose.

## VI. Experiment CA: Collision Avoidance

We conducted experiments to evaluate the behavior of the system when there are obstacles in the workspace. The parameters and parameter boundaries for the 24 joint-related parameters and six base-related parameters were the same as the ones described in Section V. The end-effector for this experiment was a custom-designed intricate-shaped 3D-printed holder for ARUCO fiducial markers [41]. The six end-effector-related parameters were obtained from the CAD design and parameter boundaries were set as $\pm 1$ cm for the translation-related parameters and $\pm 5^\circ$ for the rotation-related parameters.

The goal for the experiment was to move the end-effector, from the initial pose, to a target pose $\underline{\boldsymbol{x}}_{d,1}$ in the real task-space while avoiding collisions with the obstacles, and then insert the end-effector into a narrow slit. The target pose was obtained in real-time from a custom-designed 3D-printed holder with an ARUCO marker. To facilitate the insertion of the end-effector into the slit, we defined an intermediate pose, $\underline{\boldsymbol{x}}_{d,0}$, with the same rotation as $\underline{\boldsymbol{x}}_{d,1}$, but displaced 15 cm. The obstacles were four planes and two four-centimeter-diameter cylinders attached to a $40$ cm$^3$ cube made of aluminum frames. The position of these obstacles were obtained with respect to an ARUCO marker placed on a 3D-printed holder attached to the cube. The experimental setup is shown in Fig. 7.

Measurements of the three ARUCO markers were obtained from a camera (STC-HD853HDMI, Omron Sentech, Japan)

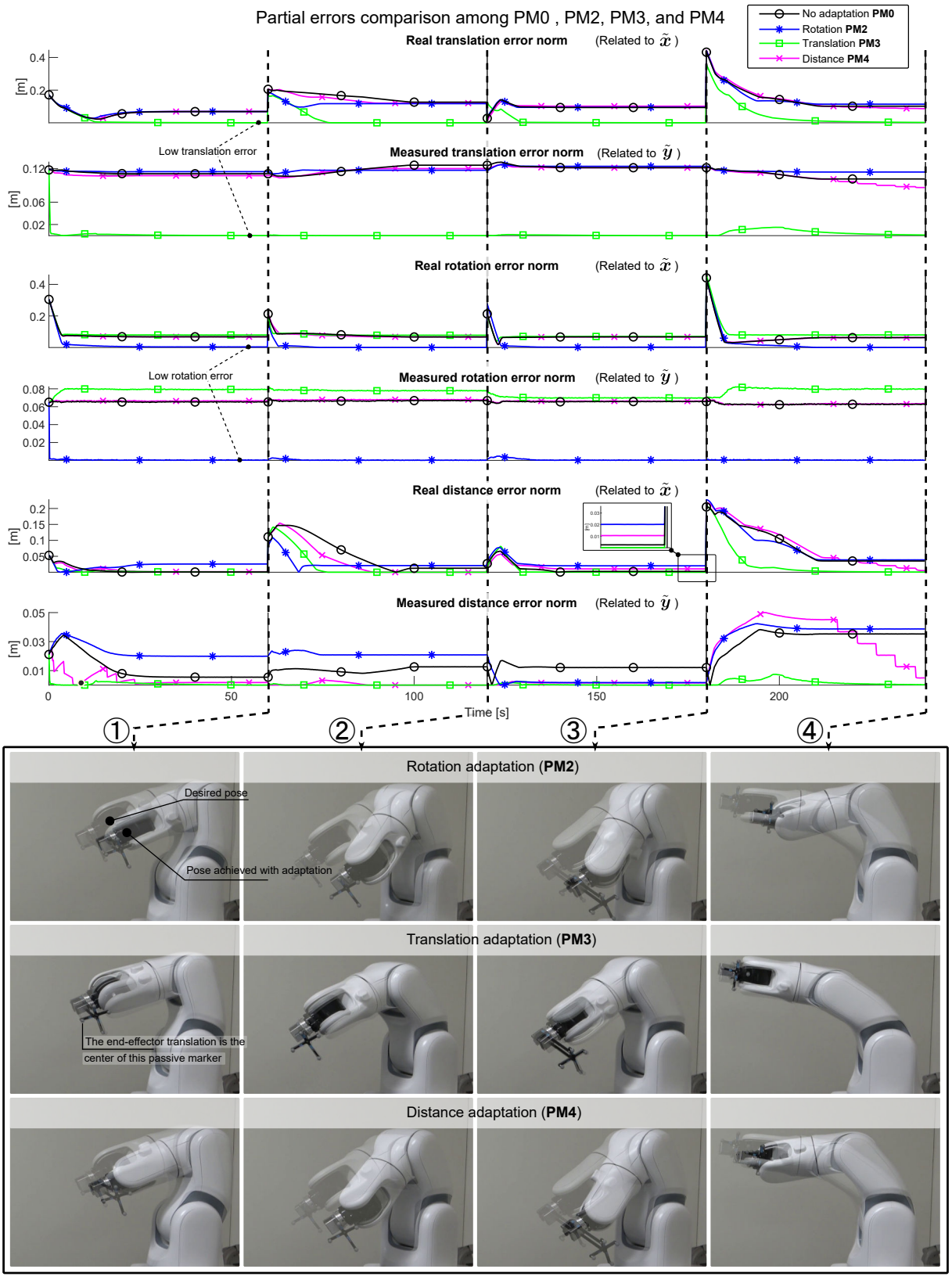


Fig. 5. Comparison of the partial errors among **PM0**, **PM2**, **PM3**, and **PM4**, in terms of real task pose error $\tilde{\boldsymbol{x}}$ and measurement pose error $\tilde{\boldsymbol{y}}$. The errors are defined in Appendix II-D. The controllers with partial measurements **PM2** and **PM3** achieve their real targets in translation and rotation, respectively, for all setpoints. The controller with partial measurements **PM4** has a distance error of several millimeters in some setpoints given that the nominal model was imprecise and the robot got stuck near a joint limit (see Fig. 6).

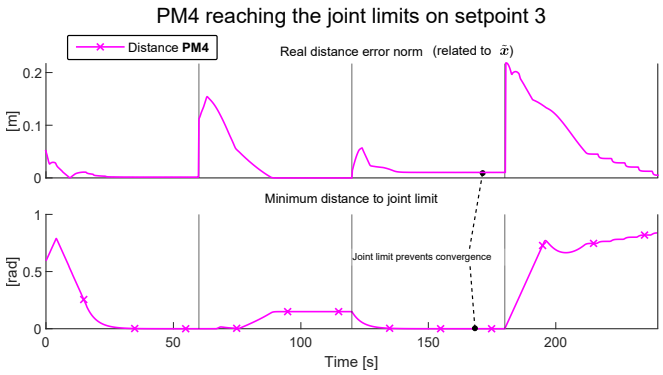


Fig. 6. The plot of the real distance error for **PM4** and the distance to a joint limit. Notice that in setpoint 3 the robot is unable to converge in distance given the proximity to a joint limit.

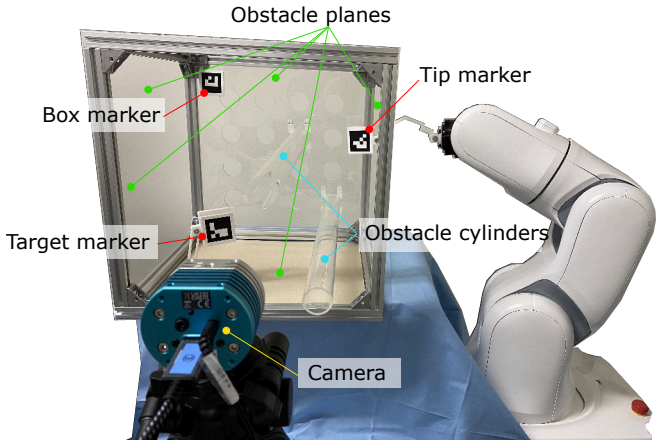


Fig. 7. Experimental setup for the collision avoidance experiment (**CA**).

and lens (VS-LDA4, Omron Sentech, Japan) set up for 1080p 60 Hz readings using a PCI-E capture board (Decklink Quad HDMI Recorder, Blackmagic Design, Australia). The camera was calibrated using MATLAB's Camera Calibration application.[9] The ARUCO recognition was implemented using OpenCV[10] at 50 Hz. The end-effector measurement was obtained online without filtering. The target pose and the cube pose were obtained using a filter based on dual-quaternion (spatial) averaging [42].

In the context of collision avoidance, the end-effector was enclosed by six spheres. This is a conservative approach, but it satisfactorily illustrates how we can use VFIs within the proposed adaptive formulation to prevent collisions between the end-effector and the obstacles in the workspace. The obstacles were two cylinders and four planes. To prevent collisions between the end-effector spheres and the obstacle cylinders, we used 12 point-to-line constraints. Those constraints require the calculation of the squared distance $D_{t_i,l_j}$ [12, Eq. (29)] between the six end-effector's spheres centered at $\boldsymbol{t}_i$, $i \in \{1,2,3,4,5,6\}$, and the two obstacle cylinders' centerlines $\underline{\boldsymbol{l}}_j$, $j \in \{1,2\}$, in addition to the Jacobians $\boldsymbol{J}_{t_i,l_j,q}$ and $\boldsymbol{J}_{t_i,l_j,\hat{\boldsymbol{a}}}$ [12, Eq. (32)] that satisfy

[9]https://www.mathworks.com/help/vision/camera-calibration.html
[10]https://docs.opencv.org/4.x/d9/d53/aruco_8hpp.html

**Algorithm 1** Proposed task-space constrained adaptive control strategy implemented in Section VI.

$T \leftarrow$ sampling time
$\underline{\boldsymbol{x}}_{\text{target}} \leftarrow$ filtered target measurement
$\underline{\boldsymbol{x}}_{\text{inter}} \leftarrow \text{getIntermediateTarget}\left(\underline{\boldsymbol{x}}_{\text{target}}\right)$
$\underline{\boldsymbol{x}}_{\text{box}} \leftarrow$ filtered box measurement
$\hat{\boldsymbol{a}} \leftarrow$ initially estimated parameters
**while** not $\text{isPlausible}\left(\boldsymbol{q}, \hat{\boldsymbol{a}}, \underline{\boldsymbol{x}}_{\text{box}}\right)$ **do**
  $\hat{\boldsymbol{a}} \leftarrow U\left(\hat{\boldsymbol{a}}_{\min}, \hat{\boldsymbol{a}}_{\max}\right)$
**end while**
**for** $\underline{\boldsymbol{x}}_d$ in $\left\{\underline{\boldsymbol{x}}_{\text{inter}}, \underline{\boldsymbol{x}}_{\text{target}}\right\}$ **do**
  **while** not $\text{stopCriterion}()$ **do**
    $\boldsymbol{q} \leftarrow$ current robot configuration
    $\breve{\boldsymbol{x}} \leftarrow \text{getTaskError}\left(\boldsymbol{q}, \hat{\boldsymbol{a}}, \underline{\boldsymbol{x}}_d\right)$
    $\boldsymbol{J}_{\hat{x},q} \leftarrow \text{getJacobianQ}\left(\boldsymbol{q}, \hat{\boldsymbol{a}}\right)$
    $\left(\boldsymbol{W}_q, \boldsymbol{w}_q\right) \leftarrow \text{getLimitsQ}\left(\boldsymbol{q}, \boldsymbol{q}_{\min}, \boldsymbol{q}_{\max}\right)$
    $\left(\boldsymbol{B}_q, \boldsymbol{b}_q\right) \leftarrow \text{getVFIsQ}\left(\boldsymbol{q}, \hat{\boldsymbol{a}}, \underline{\boldsymbol{x}}_{\text{box}}\right)$
    ▷ For the control law, see (6)
    $\boldsymbol{u}_q \leftarrow \text{controlLaw}\left(\breve{\boldsymbol{x}}, \boldsymbol{J}_{\hat{x},q}, \boldsymbol{B}_q, \boldsymbol{b}_q, \boldsymbol{W}_q, \boldsymbol{w}_q\right)$
    $\boldsymbol{y} \leftarrow$ current task-space measurement
    **if** $\boldsymbol{y}$ is valid **then**
      $\tilde{\boldsymbol{y}} \leftarrow \text{getMeasurementError}\left(\boldsymbol{q}, \hat{\boldsymbol{a}}, \boldsymbol{y}\right)$
      $\left(\boldsymbol{J}_{\hat{x},\hat{a}}, \boldsymbol{J}_{\hat{y},\hat{a}}\right) \leftarrow \text{getJacobiansA}\left(\boldsymbol{q}, \hat{\boldsymbol{a}}\right)$
      $\boldsymbol{N}_{\hat{a}} \leftarrow \text{getParametricJacobianProjector}\left(\boldsymbol{q}, \hat{\boldsymbol{a}}\right)$
      $\left(\boldsymbol{W}_{\hat{a}}, \boldsymbol{w}_{\hat{a}}\right) \leftarrow \text{getLimitsA}\left(\hat{a}, \hat{a}_{\min}, \hat{a}_{\max}\right)$
      $\left(\boldsymbol{B}_{\hat{a}}, \boldsymbol{b}_{\hat{a}}\right) \leftarrow \text{getVFIsA}\left(\boldsymbol{q}, \hat{\boldsymbol{a}}, \underline{\boldsymbol{x}}_{\text{box}}\right)$
      ▷ For the adaptation law, see (12)
      $\boldsymbol{u}_{\hat{a}} \leftarrow \text{adaptationLaw}(\breve{\boldsymbol{x}}, \tilde{\boldsymbol{y}}, \boldsymbol{J}_{\hat{x},\hat{a}}, \boldsymbol{J}_{\hat{y},\hat{a}}$
        $\boldsymbol{B}_{\hat{a}}, \boldsymbol{b}_{\hat{a}}, \boldsymbol{W}_{\hat{a}}, \boldsymbol{w}_{\hat{a}})$
      $\hat{\boldsymbol{a}} \leftarrow \hat{\boldsymbol{a}} + T\boldsymbol{u}_{\hat{a}}$
    **end if**
    $\boldsymbol{q} \leftarrow \boldsymbol{q} + T\boldsymbol{u}_q$
    $\text{sendToRobot}\left(\boldsymbol{q}\right)$
    $\text{sleepUntilNextLoop}\left()\right.$
  **end while**
**end for**

$\boldsymbol{J}_{t_i,l_j,q}\dot{\boldsymbol{q}} + \boldsymbol{J}_{t_i,l_j,\hat{\boldsymbol{a}}}\dot{\hat{\boldsymbol{a}}} = \dot{D}_{t_i,l_j}$. To prevent collisions between the end effector and the walls, we used 24 point-to-plane constraints using the distance $d_{t_i,\pi_k}$ [12, Eq. (57)] between the six end-effector spheres and the four obstacle planes $\underline{\boldsymbol{\pi}}_k$, $k \in \{1,2,3,4\}$, in addition to the Jacobians $\boldsymbol{J}_{t_i,\pi_k,q}$ and $\boldsymbol{J}_{t_i,\pi_k,\hat{a}}$ [12, Eq. (59)] that satisfy $\boldsymbol{J}_{t_i,\pi_k,q}\dot{\boldsymbol{q}} + \boldsymbol{J}_{t_i,\pi_k,\hat{\boldsymbol{a}}}\dot{\hat{\boldsymbol{a}}} = \dot{d}_{t_i,\pi_k}$. With these definitions, the VFI constraint in Problem 6 is composed of the following 36 inequalities

$$\overbrace{\begin{bmatrix} -\boldsymbol{J}_{t_1,l_1,q} \\ \vdots \\ -\boldsymbol{J}_{t_6,l_2,q} \\ -\boldsymbol{J}_{t_1,\pi_1,q} \\ \vdots \\ -\boldsymbol{J}_{t_6,\pi_4,q} \end{bmatrix}}^{\boldsymbol{B}_q} \dot{\boldsymbol{q}} \preceq \eta_{\text{vfi},q} \underbrace{\overbrace{\begin{bmatrix} D_{t_1,l_1} - D_{\text{safe},t_1,l} \\ \vdots \\ D_{t_6,l_2} - D_{\text{safe},t_6,l} \\ d_{t_1,\pi_1} - d_{\text{safe},t_1,\pi} \\ \vdots \\ d_{t_6,\pi_4} - d_{\text{safe},t_6,\pi} \end{bmatrix}}^{\boldsymbol{h}}}_{\boldsymbol{b}_q}$$

and the VFI constraint in Problem 12 is composed of the

following 36 inequalities

$$\overbrace{\begin{bmatrix} -\boldsymbol{J}_{t_1,l_1,\hat{a}} \\ \vdots \\ -\boldsymbol{J}_{t_6,l_2,\hat{a}} \\ -\boldsymbol{J}_{t_1,\pi_1,\hat{a}} \\ \vdots \\ -\boldsymbol{J}_{t_6,\pi_4,\hat{a}} \end{bmatrix}}^{\boldsymbol{B}_{\hat{a}}} \dot{\hat{\boldsymbol{a}}} \preceq \overbrace{\eta_{\text{vfi},\hat{a}} \boldsymbol{h}}^{\boldsymbol{b}_{\hat{a}}},$$

where $D_{\text{safe},t_i,l} = (R_{t_i} + R_l)^2$, with $R_{t_i} \in \{0.04, 0.015, 0.015, 0.015, 0.015, 0.075\}$ m and $R_l = 0.02$ m, and $d_{\text{safe},t_i,\pi} = R_{t_i} + d_\pi$, with $d_\pi = 0.02$. The control gains were $\eta_q = \eta_{\hat{a}} = 4$ and the damping factors were $\boldsymbol{\Lambda}_q = 0.01\boldsymbol{I}_6$ and $\boldsymbol{\Lambda}_{\hat{a}} = 0.01\boldsymbol{I}_{36}$. The VFI gains were $\eta_{\text{vfi},q} = \eta_{\text{vfi},\hat{a}} = 10$. The joint velocity limits were set at $\pm 0.01$ rad/s to partially compensate for the relatively low sampling time of the ARUCO measurements.

The initial estimated parameters were obtained by sampling from a uniform distribution within the parameter bounds until we obtained a set of parameters that did not indicate collisions with obstacles. This is necessary because although the actual robot is not in collision with the obstacles at $t = 0$ s, the *estimated* robot might be due to the uncertainties in the nominal model. In addition, the robot was given 10 s to update the parameters without motion before moving from the initial pose to $\underline{\boldsymbol{x}}_{d,0}$. This initial adaptation allows the estimated model to get reasonably close to the real model before the robot starts moving, which simplifies the tuning of the control and adaptation gains (notice that $\eta_q = \eta_{\hat{a}}$ and $\eta_{\text{vfi},q} = \eta_{\text{vfi},\hat{a}}$). After that, the robot was moved in sequence to $\underline{\boldsymbol{x}}_{d,0}$ and $\underline{\boldsymbol{x}}_{d,1}$. The controller was executed during 150 s before moving to the next set-point. The implementation for the experiment is summarized in Algorithm 1.

*Results and discussion*

The results of the experiment in terms of real task error $\tilde{\boldsymbol{x}}$, estimated task error $\breve{\boldsymbol{x}}$, measurement error $\tilde{\boldsymbol{y}}$, and minimum distance to obstacles are summarized in Fig. 8. The minimum distanceerror is the minimum distance obtained from all 36 collision pairs and a negative value indicates a penetration. Snapshot ① stands for the beginning of the experiment, where the end-effector was outside the box with a poorly estimated initial model. For 10 seconds, the parameters were updated using the ARUCO readings without moving the robot. This caused a near convergence of the measurement error and sharply improved the estimated model. The robot was then moved while adapting the parameters until snapshot ②, where the robot converged to the first setpoint $\underline{\boldsymbol{x}}_{d,0}$. At that point, all errors show convergent behavior and the setpoint was changed. The robot used the high-quality model obtained so far to move to the second setpoint, while continuously adapting the model until snapshot ③. At that point, the ARUCO readings became invalid. From that point onward, the robot relied only on the model estimated so far. Without colliding with the workspace obstacles, the robot reached $\underline{\boldsymbol{x}}_{d,1}$ with reasonable accuracy, as shown in snapshot ④. During the entire motion, the estimated model never violated a constraint. This is specially important because only the violations of the *estimated* model are relevant to the feasibility of the controller, and by consequence, to the closed-loop stability. The real model seems to have slightly penetrated a constraint (< 2 mm), but that amount is within ARUCO's expected error margin. Moreover, the conservative nature of our VFI specification prevented any physical collision.

The results of the experiment in terms of computational time are summarized in Table IV. The time spent communicating with the robot, datalogging, etc. are abbreviated as (*Comm.*). In average, the time for computing $\boldsymbol{u}_q$ using the task controller was about 0.1 ms for a 6-DoF robot with 60 inequality constraints; the time for computing $\boldsymbol{u}_{\hat{a}}$ using the adaptation controller was 7 ms, considering 36 parameters and 108 inequality constraints; and computations unrelated to the controllers took 4.9 ms (for instance, the one-way communication with the robot takes about 2.0 ms). All computations were done within the sampling time of 20 ms, with great safety margin, in all control modes.

These results indicate that the proposed strategy is an effective way to mitigate errors in the nominal model when measurements are available. Moreover, we have shown that when measurements are unavailable, the robot can perform well with the model updated up to that point. Our reproducible experimental setup can become a benchmark for future works in the field of constrained adaptive kinematic control.

TABLE IV

COMPUTATIONAL TIME FOR THE EXPERIMENT IN SECTION VI.

| | Mean [ms] | Std [ms] |
|---|---|---|
| Adaptive + Comm. ($t_{\text{AC}}$) | 11.9 | 0.32 |
| Adaptive + Task + Comm. ($t_{\text{ATC}}$) | 12.0 | 0.20 |
| Task + Comm. ($t_{\text{TC}}$) | 5.0 | 0.24 |
| Adaptive: $t_{\text{ATC}} - t_{\text{TC}}$ | 7 | - |
| Task: $t_{\text{ATC}} - t_{\text{AC}}$ | 0.1 | - |

The time spent communicating with the robot, datalogging, etc. are abbreviated as (Comm.). In terms of average behavior, the task controller took about 0.1 ms for a 6-DoF robot with 60 inequality constraints; the adaptation controller took 7 ms, for 36 parameters and 108 inequality constraints; and computations unrelated to the controllers took 4.9 ms (for instance, the one-way communication with the robot takes about 2.0 ms). All computations were done within the sampling time of 20 ms, with great safety margin, in all control modes.

## VII. CONCLUSIONS

In this work, we proposed an adaptive constrained kinematic control strategy for robots with arbitrary geometry. Our strategy is based on solving two quadratic-programming problems to generate the instantaneous control inputs. The first one, the task-space control law, reduces the estimated task error and the second one, the adaptation law, reduces

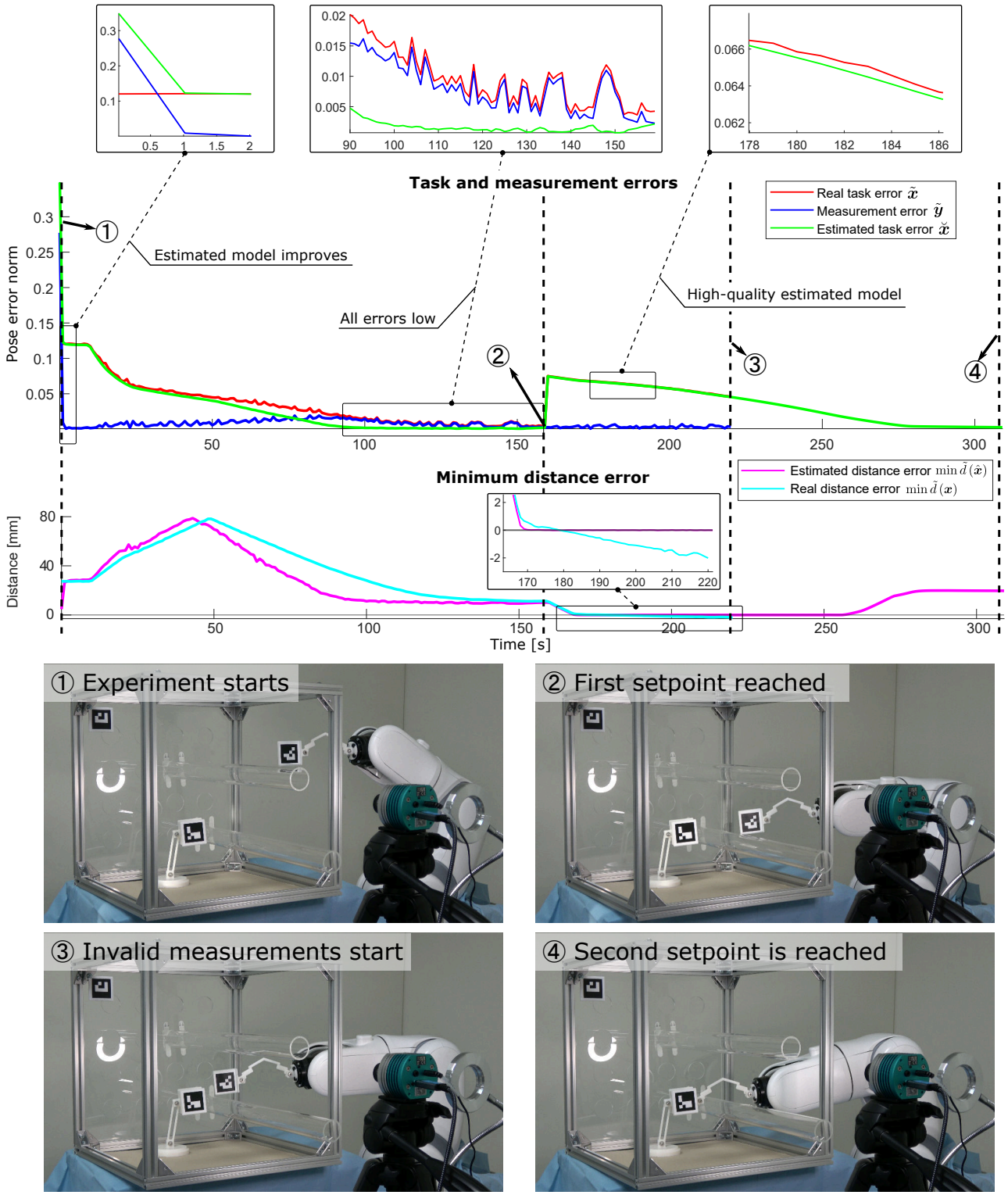


Fig. 8. Results of the experiment described in Section VI in terms of real task error $\tilde{\boldsymbol{x}}$, estimated task error $\breve{\boldsymbol{x}}$, measurement error $\tilde{\boldsymbol{y}}$, and minimum distance to obstacles. The minimum distance error is the minimum distance obtained from all 36 collision pairs and a negative value indicates a penetration. Snapshot ① stands for the beginning of the experiment, where the end-effector was outside the box with a poorly estimated initial model. For 10 seconds, the parameters were updated using the ARUCO readings without moving the robot. This caused a near convergence of the measurement error and sharply improved the estimated model. The robot was then moved while adapting the parameters until snapshot ②, where the robot converged to the first setpoint $\underline{\boldsymbol{x}}_{d,0}$. At that point, all errors show convergent behavior and the setpoint was changed. The robot used the high-quality model obtained so far to move to the second setpoint, while continuously adapting the model until snapshot ③. At that point, the ARUCO readings became invalid. From that point onward, the robot relied only on the model estimated so far. Without colliding with the workspace obstacles, the robot reached $\underline{\boldsymbol{x}}_{d,1}$ with reasonable accuracy, as shown in snapshot ④. During the entire motion, the estimated model never violated a constraint. The real model seems to have slightly penetrated a constraint (< 2 mm), but that amount is within ARUCO's expected error margin. Moreover, the conservative nature of our VFI specification prevented any physical collision.

the measurement error. We have shown that the closed-loop system under those two laws is Lyapunov stable.

Experiments have shown that, even if starting from only a rough offline calibration, our control strategy is effective when using a measurement system in the sense that the error between the estimated end-effector pose and the desired pose is always non-increasing, as our theoretical analyses predict. Also, the end-effector measurement error and the real task error always tend to improve, even when only partial measurements are available, whereas all errors in the component related to the measured variable also tend to improve. Errors in the components related to the unmeasured parts of the task-space, however, may increase compared to the baseline (i.e., when there is no task-space measurement and the control is done using the nominal forward kinematics) even if the overall errors in the task-space decrease. This is because even though we ensure that the adaptation law does not change the task vector in the direction of unmeasured components, the robot may execute different trajectories when different components are measured. Therefore, when compared to the baseline, it may be the case that the errors along unmeasured components increase.

The experimental results also have shown that our control strategy is capable of handling VFIs both in the task-space control law and adaptation law, which enables the inclusion of nonlinear constraints in the task-space variables as linear differential inequalities in the control inputs. Therefore, we can ensure that geometrical constraints in the workspace are respected while the robot parameters are adapted, increasing the overall safety.

The combination of controller parameters, such as task-space error gains and adaptation gains determine the ratio between task-space convergence and adaptation. The VFI gains and the parameter $\alpha$ that is used to split the VFIs between the task-space and adaptive control laws determine how fast the system is allowed to approach constraints, such as obstacles and joint limits, while accounting for the adaptation. Although they should be chosen in a way that prevents the robot from moving too fast before the model is properly adapted, the overall system behaves well for a wide range of values and combinations. For instance, trivial choices such as $\alpha = 0.5$, which means that the VFI gains are the same for both task-space and adaptation control laws, are suitable for real applications. Nonetheless, a more systematic procedure for tuning the parameters will be investigated in the future.

Future works will also focus on adaptively changing the safety distances of the VFIs according to the uncertainties associated with the measurement error in the adaptation law to formally ensure that constraints are not violated in the real task-space, without being overly conservative (i.e, making safe distances much larger than necessary). Also, we will extend our method to account for the second-order differential kinematics and the full robot dynamics using the (constrained) Euler-Lagrange equations.

## APPENDIX I
## PROOF OF LEMMA 4

Since $\|\boldsymbol{t}\| = \|\boldsymbol{t}_n\| = R$ and $\left\|\hat{\boldsymbol{t}}\right\| \geq R$, then $\left\|\hat{\boldsymbol{t}}\right\| = \beta R$ and $\|\boldsymbol{t}_\lambda\| = \alpha R$ with $\beta \geq \alpha \geq 1$. Hence, $\boldsymbol{t}_n = \hat{\boldsymbol{t}}/\beta$ and $\boldsymbol{t}_\lambda = \hat{\boldsymbol{t}}\left(\beta - \beta\lambda + \lambda\right)/\beta$. Thus, $\|\boldsymbol{t}_\lambda\| = \left(\beta - \beta\lambda + \lambda\right)R$, which implies $\alpha = \beta - \beta\lambda + \lambda$ and $\boldsymbol{t}_\lambda = \hat{\boldsymbol{t}}\left(\alpha/\beta\right)$.

Let $D_{\hat{t}t} \triangleq \left\|\hat{\boldsymbol{t}} - \boldsymbol{t}\right\|^2$ and $D_{\lambda t} = \|\boldsymbol{t}_\lambda - \boldsymbol{t}\|^2$. Since $D_{\hat{t}t} \geq D_{\lambda t}$ implies $\left\|\hat{\boldsymbol{t}} - \boldsymbol{t}\right\| \geq \|\boldsymbol{t}_\lambda - \boldsymbol{t}\|$, it suffices to show that $D_{\hat{t}t} \geq D_{\lambda t}$. Let us assume, for the sake of contradiction, that $D_{\hat{t}t} < D_{\lambda t}$. Since $D_{\hat{t}t} = \left\|\hat{\boldsymbol{t}}\right\|^2 - 2\boldsymbol{t}^T\hat{\boldsymbol{t}} + \|\boldsymbol{t}\|^2$ and $D_{\lambda t} = \|\boldsymbol{t}_\lambda\|^2 - 2\boldsymbol{t}^T\boldsymbol{t}_\lambda + \|\boldsymbol{t}\|^2$, then $(\beta R)^2 - 2\boldsymbol{t}^T\hat{\boldsymbol{t}} + R^2 < (\alpha R)^2 - 2\boldsymbol{t}^T\hat{\boldsymbol{t}}\left(\alpha/\beta\right) + R^2$, which implies $(\beta^2 - \alpha^2)R^2 < 2\boldsymbol{t}^T\hat{\boldsymbol{t}}\left[(\beta - \alpha)/\beta\right]$ and thus $(\beta + \alpha)R^2 < 2\boldsymbol{t}^T\hat{\boldsymbol{t}}/\beta$. Because $\boldsymbol{t}^T\hat{\boldsymbol{t}} = \|\boldsymbol{t}\|\left\|\hat{\boldsymbol{t}}\right\|\cos\phi_{t\hat{t}}$, we obtain $\beta + \alpha < 2\cos\phi_{t\hat{t}}$. Since $(2\cos\phi_{t\hat{t}}) \in [-2, 2]$, we conclude that $\beta + \alpha < 2$, which is a contradiction because $\beta + \alpha \geq 2$. Therefore, it is not true that $D_{\hat{t}t} < D_{\lambda t}$. Hence, $D_{\hat{t}t} \geq D_{\lambda t}$, which concludes the proof.

## APPENDIX II
## PARAMETERIZATION OF RIGID MOTIONS, ROTATIONS, AND TRANSLATIONS

The use cases explored in this work have been defined in general terms in Section IV. Nonetheless, their implementation depends on the choice of parameterization. We use dual quaternion algebra to represent rigid motion transformations, in which translations and rotations are particular cases. Since our general formulation is not dependent on any particular parameterization, here we do not present any comparison of dual quaternion algebra with other representations. Instead, the goal of this appendix is to ensure completeness of presentation and reproducibility in case readers want to replicate our results, which would require using the same representation that we have used. Interested readers can find a gentle introduction to dual quaternion algebra in [43]. A discussion about its advantages in the context of constrained control can be found in [12]. The computational library that we use to manipulate elements of dual quaternion algebra is described in [40].

Let the quaternion set be

$$\mathbb{H} \triangleq \left\{h_1 + \hat{i}h_2 + \hat{j}h_3 + \hat{k}h_4 \,:\, h_1, h_2, h_3, h_4 \in \mathbb{R}\right\}$$

with $\hat{i}^2 = \hat{j}^2 = \hat{k}^2 = \hat{i}\hat{j}\hat{k} = -1$ and the dual quaternion set be

$$\mathcal{H} \triangleq \left\{\boldsymbol{h} + \varepsilon\boldsymbol{h}' : \boldsymbol{h}, \boldsymbol{h}' \in \mathbb{H},\, \varepsilon^2 = 0,\, \varepsilon \neq 0\right\}.$$

### A. Positions and translations

The set of pure quaternions is given by $\mathbb{H}_p \triangleq \{\boldsymbol{h} \in \mathbb{H} : \mathrm{Re}\left(\boldsymbol{h}\right) = 0\}$, where $\mathrm{Re}\left(h_1 + \hat{i}h_2 + \hat{j}h_3 + \hat{k}h_4\right) = h_1$, and is isomorphic to $\mathbb{R}^3$ under the addition operation.

### B. Orientations and rotations

The unit quaternion set, $\mathbb{S}^3 \triangleq \{\boldsymbol{h} \in \mathbb{H} : \|\boldsymbol{h}\| = 1\}$, contains elements that represent orientations and rotations in the tridimensional space. When the set $\mathbb{S}^3$ is equipped with the multiplication operation, we obtain the group $\mathrm{Spin}(3)$ of rotations, which double covers $\mathrm{SO}(3)$.

### C. Poses and rigid motions

Elements of the unit dual quaternion set, $\underline{\boldsymbol{\mathcal{S}}} \triangleq \left\{\boldsymbol{r} + (1/2)\,\varepsilon \boldsymbol{t}\boldsymbol{r} : \boldsymbol{r} \in \mathbb{S}^3, \boldsymbol{t} \in \mathbb{H}_p\right\} \subset \mathcal{H}$, represent poses and rigid motions in the tridimensional space. Analogously to $\mathrm{Spin}(3)$, when $\underline{\boldsymbol{\mathcal{S}}}$ is equipped with the multiplication operation, we obtain the group $\mathrm{Spin}(3) \ltimes \mathbb{R}^3$, which double covers $\mathrm{SE}(3)$.

### D. Error definition

TABLE V

COMPLETE AND PARTIAL TASK-SPACE VALUES.

| | Estimated |
|---|---|
| $\underline{\boldsymbol{y}}, \boldsymbol{y}_r, \boldsymbol{y}_t, y_d$ | Measured pose, rotation, translation, and distance, respectively. |
| $\underline{\boldsymbol{x}}_d, \boldsymbol{r}_d, \boldsymbol{t}_d, d_d$ | Desired pose, rotation, translation, and distance, respectively. |
| $\hat{\underline{\boldsymbol{x}}}, \hat{\boldsymbol{r}}, \hat{\boldsymbol{t}}, \hat{d}$ | Estimated pose, rotation, translation, and distance, respectively. |
| $\underline{\boldsymbol{x}}, \boldsymbol{r}, \boldsymbol{t}, d$ | Real pose, rotation, translation, and distance, respectively. |

TABLE VI

COMPLETE AND PARTIAL ERROR DEFINITIONS.

| | Estimated | Real | Measurement |
|---|---|---|---|
| Distance | $\breve{d} = \hat{d} - d_d$ | $\tilde{d} = d - d_d$ | $\tilde{y}_d = \hat{d} - y_d$ |
| Translation | $\breve{\boldsymbol{t}} = \hat{\boldsymbol{t}} - \boldsymbol{t}_d$ | $\tilde{\boldsymbol{t}} = \boldsymbol{t} - \boldsymbol{t}_d$ | $\tilde{\boldsymbol{y}}_t = \hat{\boldsymbol{t}} - \boldsymbol{y}_t$ |
| Rotation | $\breve{\boldsymbol{r}} = \mathcal{E}(\hat{\boldsymbol{r}}, \boldsymbol{r}_d)$ | $\tilde{\boldsymbol{r}} = \mathcal{E}(\boldsymbol{r}, \boldsymbol{r}_d)$ | $\tilde{\boldsymbol{y}}_r = \mathcal{E}(\hat{\boldsymbol{r}}, \boldsymbol{y}_r)$ |
| Pose | $\breve{\underline{\boldsymbol{x}}} \triangleq \mathcal{E}(\hat{\underline{\boldsymbol{x}}}, \underline{\boldsymbol{x}}_d)$ | $\tilde{\underline{\boldsymbol{x}}} \triangleq \mathcal{E}(\underline{\boldsymbol{x}}, \underline{\boldsymbol{x}}_d)$ | $\tilde{\underline{\boldsymbol{y}}} \triangleq \mathcal{E}\left(\hat{\underline{\boldsymbol{x}}}, \underline{\boldsymbol{y}}\right)$ |

Consider a desired end-effector pose $\underline{\boldsymbol{\mathcal{S}}} \ni \underline{\boldsymbol{x}}_d = \boldsymbol{r}_d + \varepsilon\frac{1}{2}\boldsymbol{t}_d\boldsymbol{r}_d$, in which $\boldsymbol{r}_d \in \mathbb{S}^3$ is the desired end-effector orientation and $\boldsymbol{t}_d \in \mathbb{H}_p$ is the desired end-effector position. Analogously, let $\hat{\underline{\boldsymbol{x}}} \in \underline{\boldsymbol{\mathcal{S}}}$ be the estimated end-effector pose, $\underline{\boldsymbol{x}} \in \underline{\boldsymbol{\mathcal{S}}}$ be the real end-effector pose, and $\underline{\boldsymbol{y}} \in \underline{\boldsymbol{\mathcal{S}}}$ be the measured end-effector pose. These elements are summarized in Table V.

Using the aforementioned elements, the error definitions are summarized in Table VI and further explained as follows. The estimated distance error and translation error, similarly to their counterparts parameterized with elements of the Euclidean group $\left(\mathbb{R}^3, +\right)$, are defined as $\breve{d} = \hat{d} - d_d = \left\|\hat{\boldsymbol{t}}\right\|_2 - \|\boldsymbol{t}_d\|_2$ and $\breve{\boldsymbol{t}} = \hat{\boldsymbol{t}} - \boldsymbol{t}_d$, respectively. On the other hand, orientation and pose errors that respect the topology of the underlying space of orientations and rigid motions are defined by multiplications in both $\mathrm{Spin}(3)$ and $\mathrm{Spin}(3) \ltimes \mathbb{R}^3$, respectively [44]. For instance, the estimated pose error is defined as $\breve{\underline{\boldsymbol{x}}} \triangleq \mathcal{E}(\hat{\underline{\boldsymbol{x}}}, \underline{\boldsymbol{x}}_d)$ such that

$$\mathcal{E}(\hat{\underline{\boldsymbol{x}}}, \underline{\boldsymbol{x}}_d) = \begin{cases} \hat{\underline{\boldsymbol{x}}}^* \underline{\boldsymbol{x}}_d - 1 & \text{if } \|\hat{\underline{\boldsymbol{x}}}^* \underline{\boldsymbol{x}}_d - 1\|_2 < \|\hat{\underline{\boldsymbol{x}}}^* \underline{\boldsymbol{x}}_d + 1\|_2 \\ \hat{\underline{\boldsymbol{x}}}^* \underline{\boldsymbol{x}}_d + 1 & \text{otherwise,} \end{cases} \tag{21}$$

and the estimated rotation error as $\breve{\boldsymbol{r}} = \mathcal{E}(\hat{\boldsymbol{r}}, \boldsymbol{r}_d)$ [45], where $\hat{\underline{\boldsymbol{x}}}^*$ and $\hat{\boldsymbol{r}}^*$ are the dual quaternion conjugate of $\hat{\underline{\boldsymbol{x}}}$ and quaternion conjugate of $\hat{\boldsymbol{r}}$, respectively. The reason for using such pose and rotation errors is to prevent the problem of unwinding [46], because $\breve{\underline{\boldsymbol{x}}}$ and $-\breve{\underline{\boldsymbol{x}}}$ represent the same pose, and similarly $\breve{\boldsymbol{r}}$ and $-\breve{\boldsymbol{r}}$ represent the same orientation.

### E. Mapping errors to vectors

Using the bijective operators $\mathrm{vec}_3 : \mathbb{H}_p \to \mathbb{R}^3$, $\mathrm{vec}_4 : \mathbb{H} \to \mathbb{R}^4$, and $\mathrm{vec}_8 : \underline{\boldsymbol{\mathcal{S}}} \to \mathbb{R}^8$, which take the coefficients of (dual) quaternions and stack them into a vector, we map those errors to vectors to comply with the formulation in Sections II–IV.

### F. Error norms

Rewriting the estimated pose error as $\breve{\underline{\boldsymbol{x}}} = \mathcal{E}(\hat{\underline{\boldsymbol{x}}}, \underline{\boldsymbol{x}}_d) = \breve{\boldsymbol{x}}_P + \varepsilon\breve{\boldsymbol{x}}_D$, the estimated pose error norm used in Sections V–VI is defined as $\|\breve{\underline{\boldsymbol{x}}}\|_2 \triangleq \|\breve{\boldsymbol{x}}_P\|_2 + \|\breve{\boldsymbol{x}}_D\|_2$, whereas the orientation, position, and distance norms are defined as $\|\breve{\boldsymbol{r}}\|_2$, $\left\|\breve{\boldsymbol{t}}\right\|_2$, and $\left\|\breve{d}\right\|_2 = \left|\breve{d}\right|$, respectively.[11]

## APPENDIX III
## ANALYSIS OF CLOSED-LOOP STABILITY FOR MULTIPLICATIVE ERRORS

Closed-loop stability is also guaranteed when using multiplicative errors such as (21). This can be shown by following the procedure described in Section III-D.

Indeed, let the task-space be defined as the set of all end-effector's poses, parameterized using $\underline{\boldsymbol{\mathcal{S}}}$. Consider the Lyapunov function $V(\breve{\boldsymbol{x}}) = \frac{1}{2}\breve{\boldsymbol{x}}^T\breve{\boldsymbol{x}}$, where

$$\breve{\boldsymbol{x}} = \mathrm{vec}_8\left(\mathcal{E}\left(\hat{\underline{\boldsymbol{x}}}, \underline{\boldsymbol{x}}_d\right)\right). \tag{22}$$

The time derivative of $\breve{\boldsymbol{x}}$ is given by $\dot{\breve{x}} = \mathrm{vec}_8\left(\dot{\hat{\underline{\boldsymbol{x}}}}^* \underline{\boldsymbol{x}}_d\right)$ because $\dot{\underline{\boldsymbol{x}}}_d = 0$ for all $t$. Using the Hamilton operator $\bar{\boldsymbol{H}}_8 : \mathcal{H} \to \mathbb{R}^{8\times 8}$, such that $\mathrm{vec}_8(\underline{\boldsymbol{a}}\underline{\boldsymbol{b}}) = \bar{\boldsymbol{H}}_8(\underline{\boldsymbol{b}})\,\mathrm{vec}_8\,\underline{\boldsymbol{a}}$, with $\underline{\boldsymbol{a}}, \underline{\boldsymbol{b}} \in \mathcal{H}$, we obtain $\dot{\breve{x}} = \bar{\boldsymbol{H}}_8(\underline{\boldsymbol{x}}_d)\,\boldsymbol{C}_8\,\mathrm{vec}_8\,\dot{\hat{\underline{\boldsymbol{x}}}}$, where $\boldsymbol{C}_8 \in \mathbb{R}^{8\times 8}$ is the matrix that satisfies $\mathrm{vec}_8\,\underline{\boldsymbol{a}}^* = \boldsymbol{C}_8\,\mathrm{vec}_8\,\underline{\boldsymbol{a}}$ for all $\underline{\boldsymbol{a}} \in \mathcal{H}$ [43]. Defining $\dot{\hat{\boldsymbol{x}}} \triangleq \mathrm{vec}_8\,\dot{\hat{\underline{\boldsymbol{x}}}}$, we obtain $\dot{\breve{\boldsymbol{x}}} = \bar{\boldsymbol{H}}_8(\underline{\boldsymbol{x}}_d)\,\boldsymbol{C}_8\boldsymbol{J}_{\hat{x}}\dot{\boldsymbol{v}}$, where $\boldsymbol{J}_{\hat{x}}$ and $\dot{\boldsymbol{v}}$ are defined as in (16). Let $\boldsymbol{G}_{\hat{x}} \triangleq \bar{\boldsymbol{H}}_8(\underline{\boldsymbol{x}}_d)\,\boldsymbol{C}_8\boldsymbol{J}_{\hat{x}}$, then $\dot{\breve{\boldsymbol{x}}} = \boldsymbol{G}_{\hat{x}}\dot{\boldsymbol{v}}$ and

$$\dot{V}(\breve{\boldsymbol{x}}) = \breve{\boldsymbol{x}}^T\boldsymbol{G}_{\hat{x}}\dot{\boldsymbol{v}} = \breve{\boldsymbol{x}}^T\boldsymbol{G}_{\hat{x}}\boldsymbol{u}_{v,q} + \breve{\boldsymbol{x}}^T\boldsymbol{G}_{\hat{x}}\boldsymbol{u}_{v,\hat{a}}. \tag{23}$$

We partition $\boldsymbol{G}_{\hat{x}}$ such that $\boldsymbol{G}_{\hat{x}} = \begin{bmatrix}\boldsymbol{G}_{\hat{x},q} & \boldsymbol{G}_{\hat{x},\hat{a}}\end{bmatrix}$ and replace $\boldsymbol{J}_{\hat{x},q}$ with $\boldsymbol{G}_{\hat{x},q}$ in (6), where $\breve{\boldsymbol{x}}$ is defined in (22).

Now, consider two cases for the measurement space, where the end-effector pose space and the end-effector orientation space are parameterized as $\underline{\boldsymbol{\mathcal{S}}}$ and $\mathbb{S}^3$, respectively. We define $\tilde{\boldsymbol{y}} = \mathrm{vec}_8\left(\mathcal{E}\left(\hat{\underline{\boldsymbol{x}}}, \underline{\boldsymbol{x}}\right)\right)$ for the pose or $\tilde{\boldsymbol{y}} = \mathrm{vec}_4\left(\mathcal{E}\left(\hat{\boldsymbol{r}}, \boldsymbol{r}\right)\right)$ for the orientation.

Using any of those definitions, we replace $\boldsymbol{J}_{\hat{y},\hat{a}}$ with $\boldsymbol{G}_{\hat{y},\hat{a}}$ and $\boldsymbol{J}_{\hat{x},\hat{a}}$ with $\boldsymbol{G}_{\hat{x},\hat{a}}$ in (12), where $\boldsymbol{G}_{\hat{y}} = \begin{bmatrix}\boldsymbol{G}_{\hat{y},q} & \boldsymbol{G}_{\hat{y},\hat{a}}\end{bmatrix} = \bar{\boldsymbol{H}}_8(\underline{\boldsymbol{x}})\,\boldsymbol{C}_8\boldsymbol{J}_{\hat{x}}$ for the pose or $\boldsymbol{G}_{\hat{y}} = \bar{\boldsymbol{H}}_4(\boldsymbol{r})\,\boldsymbol{C}_4\boldsymbol{J}_{\hat{\boldsymbol{r}}}$ for the orientation, where $\bar{\boldsymbol{H}}_4 : \mathbb{H} \to \mathbb{R}^{4\times 4}$ satisfies $\mathrm{vec}_4(\boldsymbol{a}\boldsymbol{b}) =$

[11] The quaternion norm is equivalent to the Euclidean norm, but the dual quaternion norm is not. Therefore, if $\boldsymbol{h} \in \mathbb{H}$, then $\|\boldsymbol{h}\| = \|\boldsymbol{h}\|_2$. In contrast, given $\underline{\boldsymbol{h}} \in \mathcal{H}$, except for particular cases, usually $\|\underline{\boldsymbol{h}}\| \neq \|\underline{\boldsymbol{h}}\|_2$.

$\bar{\boldsymbol{H}}_4(\boldsymbol{b})\,\mathrm{vec}_4\,\boldsymbol{a}$, with $\boldsymbol{a}, \boldsymbol{b} \in \mathbb{H}$, and $\boldsymbol{C}_4$ satisfy $\mathrm{vec}\,\boldsymbol{r}^* = \boldsymbol{C}_4\,\mathrm{vec}_4\,\boldsymbol{r}$.

Therefore, by replacing $\boldsymbol{J}$ with $\boldsymbol{G}$ (using the appropriate subscripts), the analysis is essentially the same, and thus we conclude that multiplicative errors such as (22) do not affect closed-loop stability.

## REFERENCES


[1] B. Siciliano, L. Sciavicco, L. Villani, and G. Oriolo, *Robotics: Modelling, Planning and Control*, ser. Advanced Textbooks in Control and Signal Processing. London: Springer-Verlag London, 2009.

[2] R. Lenz and R. Tsai, "Calibrating a cartesian robot with eye-on-hand configuration independent of eye-to-hand relationship," in *Proceedings CVPR88: The Computer Society Conference on Computer Vision and Pattern Recognition*. IEEE Comput. Soc. Press.

[3] B. W. Mooring, Z. S. Roth, and M. R. Driels, *Fundamentals of manipulator calibration*. Wiley New York, 1991.

[4] J. M. S. Motta, G. C. de Carvalho, and R. McMaster, "Robot calibration using a 3d vision-based measurement system with a single camera," *Robotics and Computer-Integrated Manufacturing*, vol. 17, no. 6, pp. 487–497, dec 2001.

[5] X. Zhang, Y. Song, Y. Yang, and H. Pan, "Stereo vision based autonomous robot calibration," *Robotics and Autonomous Systems*, vol. 93, pp. 43–51, jul 2017.

[6] M. M. Marinho, K. Harada, A. Morita, and M. Mitsuishi, "Smartarm: Integration and validation of a versatile surgical robotic system for constrained workspaces," *The International Journal of Medical Robotics and Computer Assisted Surgery (IJMRCAS)*, vol. 16, no. 2, p. e2053, Apr. 2020.

[7] M. M. Marinho, K. Harada, K. Deie, T. Ishimaru, and M. Mitsuishi, "SmartArm: Suturing feasibility of a surgical robotic system on a neonatal chest model," *IEEE Transactions on Medical Robotics and Bionics*, vol. 3, no. 1, pp. 253–256, feb 2021.

[8] Z. Wang, Z. Liu, Q. Ma, A. Cheng, Y. hui Liu, S. Kim, A. Deguet, A. Reiter, P. Kazanzides, and R. H. Taylor, "Vision-based calibration of dual RCM-based robot arms in human-robot collaborative minimally invasive surgery," *IEEE Robotics and Automation Letters*, vol. 3, no. 2, pp. 672–679, apr 2018.

[9] M. Yoshimura, M. M. Marinho, K. Harada, and M. Mitsuishi, "Single shot pose estimation of surgical robot instruments' shafts from monocular endoscopic images," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, May 2020, pp. 9960–9966.

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

[11] A. Escande, N. Mansard, and P.-B. Wieber, "Hierarchical quadratic programming: Fast online humanoid-robot motion generation," *The International Journal of Robotics Research*, vol. 33, no. 7, pp. 1006–1028, may 2014. [Online]. Available: http://ijr.sagepub.com/cgi/doi/10.1177/0278364914521306

[12] M. M. Marinho, B. V. Adorno, K. Harada, and M. Mitsuishi, "Dynamic active constraints for surgical robots using vector-field inequalities," *IEEE Transactions on Robotics*, vol. 35, no. 5, pp. 1166–1185, Oct. 2019.

[13] J. J. Quiroz-Omana and B. V. Adorno, "Whole-body control with (self) collision avoidance using vector field inequalities," *IEEE Robotics and Automation Letters*, vol. 4, no. 4, pp. 4048–4053, oct 2019.

[14] S. Gharaaty, T. Shu, A. Joubair, W. F. Xie, and I. A. Bonev, "Online pose correction of an industrial robot using an optical coordinate measure machine system," *International Journal of Advanced Robotic Systems*, vol. 15, no. 4, p. 172988141878791, jul 2018.

[15] C. Yu and J. Xi, "Simultaneous and on-line calibration of a robot-based inspecting system," *Robotics and Computer-Integrated Manufacturing*, vol. 49, pp. 349–360, feb 2018.

[16] V. Bonnet, K. Pfeiffer, P. Fraisse, A. Crosnier, and G. Venture, "Self-generation of optimal exciting motions for identification of a humanoid robot," *International Journal of Humanoid Robotics*, vol. 15, no. 06, p. 1850024, dec 2018.

[17] T. Katsumata, B. Navarro, V. Bonnet, P. Fraisse, A. Crosnier, and G. Venture, "Optimal exciting motion for fast robot identification. application to contact painting tasks with estimated external forces," *Robotics and Autonomous Systems*, vol. 113, pp. 149–159, mar 2019.

[18] A. P. Morgan and K. S. Narendra, "On the uniform asymptotic stability of certain linear nonautonomous differential equations," *SIAM Journal on Control and Optimization*, vol. 15, no. 1, pp. 5–24, jan 1977.

[19] J.-J. E. Slotine, W. Li *et al.*, *Applied nonlinear control*. Prentice hall Englewood Cliffs, NJ, 1991, vol. 199, no. 1.

[20] J.-J. E. Slotine and W. Li, "On the adaptive control of robot manipulators," *The International Journal of Robotics Research*, vol. 6, no. 3, pp. 49–59, sep 1987.

[21] G. Garofalo, X. Wu, and C. Ott, "Adaptive passivity-based multi-task tracking control for robotic manipulators," *IEEE Robotics and Automation Letters*, vol. 6, no. 4, pp. 7129–7136, oct 2021.

[22] C. C. Cheah, C. Liu, and J. J. E. Slotine, "Adaptive tracking control for robots with unknown kinematic and dynamic properties," *The International Journal of Robotics Research*, vol. 25, no. 3, pp. 283–296, mar 2006.

[23] F. Lewis, *Robot manipulator control : theory and practice*. New York: Marcel Dekker, 2004.

[24] J. Yuan and B. Yuan, "Recursive computation of the slotine-li regressor," in *Proceedings of 1995 American Control Conference - ACC'95*. American Autom Control Council.

[25] C. Cheah, C. Liu, and J. Slotine, "Approximate jacobian adaptive control for robot manipulators," in *IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA '04. 2004*. IEEE, 2004.

[26] T. Marcucci, C. Della Santina, M. Gabiccini, and A. Bicchi, "Towards minimum-information adaptive controllers for robot manipulators," in *2017 American Control Conference (ACC)*. IEEE, 2017, pp. 4209–4214.

[27] T. Osa, J. Pajarinen, G. Neumann, J. A. Bagnell, P. Abbeel, and J. Peters, "An algorithmic perspective on imitation learning," *Foundations and Trends in Robotics*, vol. 7, no. 1-2, pp. 1–179, 2018.

[28] M. Okada and T. Taniguchi, "Dreaming: Model-based reinforcement learning by latent imagination without reconstruction," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, may 2021.

[29] J. Fischer, C. Eyberg, M. Werling, and M. Lauer, "Sampling-based inverse reinforcement learning algorithms with safety constraints," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, sep 2021.

[30] M. N. Finean, W. Merkt, and I. Havoutis, "Simultaneous scene reconstruction and whole-body motion planning for safe operation in dynamic environments," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, sep 2021.

[31] S. Hu, E. Babaians, M. Karimi, and E. Steinbach, "NMPC-MP: Real-time nonlinear model predictive control for safe motion planning in manipulator teleoperation," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, sep 2021.

[32] J. Wu, Z. Jin, A. Liu, L. Yu, and F. Yang, "A survey of learning-based control of robotic visual servoing systems," *Journal of the Franklin Institute*, vol. 359, no. 1, pp. 556–577, jan 2022.

[33] H. Shi, G. Sun, Y. Wang, and K.-S. Hwang, "Adaptive image-based visual servoing with temporary loss of the visual signal," *IEEE Transactions on Industrial Informatics*, vol. 15, no. 4, pp. 1956–1965, apr 2019.

[34] Y. Zhang and S. Li, "A neural controller for image-based visual servoing of manipulators with physical constraints," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 29, no. 11, pp. 5419–5429, nov 2018.

[35] F. Sadeghi, A. Toshev, E. Jang, and S. Levine, "Sim2real viewpoint invariant visual servoing by recurrent control," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*. IEEE, jun 2018.

[36] R. Tsai and R. Lenz, "A new technique for fully autonomous and efficient 3d robotics hand/eye calibration," *IEEE Transactions on Robotics and Automation*, vol. 5, no. 3, pp. 345–358, jun 1989.

[37] F. Zhong, Z. Wang, W. Chen, K. He, Y. Wang, and Y.-H. Liu, "Hand-eye calibration of surgical instrument for robotic surgery using interactive manipulation," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 1540–1547, apr 2020.

[38] T. H. Gronwall, "Note on the Derivatives with Respect to a Parameter of the Solutions of a System of Differential Equations," *The Annals of Mathematics*, vol. 20, no. 4, p. 292, jul 1919.

[39] V. M. Goncalves, B. V. Adorno, A. Crosnier, and P. Fraisse, "Stable-by-Design Kinematic Control Based on Optimization," *IEEE*

*Transactions on Robotics*, vol. 36, no. 3, pp. 644–656, jun 2020. [Online]. Available: https://ieeexplore.ieee.org/document/8962953/

[40] B. V. Adorno and M. Marques Marinho, "DQ Robotics: A Library for Robot Modeling and Control," *IEEE Robotics & Automation Magazine*, vol. 28, no. 3, pp. 102–116, sep 2021. [Online]. Available: https://ieeexplore.ieee.org/document/9136790/ http://arxiv.org/abs/1910.11612

[41] S. Garrido-Jurado, R. Muñoz-Salinas, F. Madrid-Cuevas, and M. Marín-Jiménez, "Automatic generation and detection of highly reliable fiducial markers under occlusion," *Pattern Recognition*, vol. 47, no. 6, pp. 2280–2292, jun 2014.

[42] B. V. Adorno, "Manipulação Cooperativa Descentralizada Usando o Espaço Dual de Cooperação," in *XIX Congresso Brasileiro de Automática*. Campina Grande: Sociedade Brasileira de Automática, 2012, pp. 1436–1443.

[43] ——, "Robot kinematic modeling and control based on dual quaternion algebra — Part I: Fundamentals," 2017.

[44] H. Pham, B. Adorno, V. Perdereau, and P. Fraisse, "Set-point control of robot end-effector pose using dual quaternion feedback," *Robotics and Computer-Integrated Manufacturing*, vol. 52, pp. 100–110, aug 2018.

[45] M. M. Marinho, B. V. Adorno, K. Harada, K. Deie, A. Deguet, P. Kazanzides, R. H. Taylor, and M. Mitsuishi, "A unified framework for the teleoperation of surgical robots in constrained workspaces," in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, may 2019.

[46] H. T. Kussaba, L. F. Figueredo, J. Y. Ishihara, and B. V. Adorno, "Hybrid kinematic control for rigid body pose stabilization using dual quaternions," *Journal of the Franklin Institute*, vol. 354, no. 7, pp. 2769–2787, may 2017.