Interlinked Visual Tracking and Robotic Manipulation of Articulated Objects

Antonio Paolillo, Kevin Chappellet, Anastasia Bolotnikova, Abderrahmane Kheddar

Interlinked Visual Tracking and Robotic Manipulation of Articulated Objects

Antonio Paolillo, Kévin Chappellet, Anastasia Bolotnikova, Abderrahmane Kheddar

***Abstract*—Robotic manipulation tasks require the knowledge on the configuration of the object in use. Since most objects are generally not equipped with any sensor, an estimator is required. Furthermore, if an object is articulated, i.e. includes passive joints, the estimation process has to reconstruct the pose of the object floating base and its joints variables, concurrently with the manipulation control. We address the estimation problem with an online virtual visual servoing paradigm written as a quadratic program. Our estimator is integrated in closed-loop with the manipulation control governing the robot, which is also a quadratic program. Tracking and manipulation experiments, using a humanoid, show the effectiveness of our approach.**

***Index Terms*—Sensor-based Control, Visual Tracking, Perception for Grasping and Manipulation.**

I. Introduction

In the perspective of having domestic or office robots manipulating and interacting with articulated human-tailored objects, such as households appliances, they shall be embedded with perception capabilities allowing to monitor the state of such objects, whose models are usually available. For example, to manipulate the drawer of a dresser (see Fig. 1), a robot needs the knowledge of the handle pose to grasp it, and the drawer joint position to control the opening motion.

In doing so, the estimation and the manipulation processes have to be jointly integrated with the whole body control of the robot, to be consistent with other motion constraints such as (auto)collision avoidance, joint torque and state limits, etc.

The original motivation behind this work (beyond its other potential use), is to close the loop on a multi-objective task-space controller formulated as a quadratic program (QP) that models the manipulated object as an augmentation of the robot structure [1], [2]. Indeed, robot(s) and manipulated objects are integrated in a "multi-robot system" that is controlled with a single QP when they come to interact. Subsequently, the decision variables of classical QP frameworks controlling a single robot –that are the generalized joint acceleration and contact forces, are augmented by those of the manipulated (articulated) objects resulting in a multi-robot QP (MQP) control framework. The MQP requires the current state value of the overall multi-robot system to close the loop and update the models used in the tasks and constraints. Unless they are other robots, the manipulated articulated objects are not equipped with sensors measuring their configuration. Therefore, since the robot can be embedded with a camera, we first considered existing articulated objects tracking methods that reveal limitations we discuss in Section II. We devised a configuration estimator as a virtual visual servoing in Sect. III and formulate it as a QP (Sect. IV), using proper visual features as described in Sect. V. This allows integrating constraints on the estimation variable, and the implementation tools from the MQP control framework, see experiments in Sect. VI. Limitations and perspectives are discussed in Sect. VII.

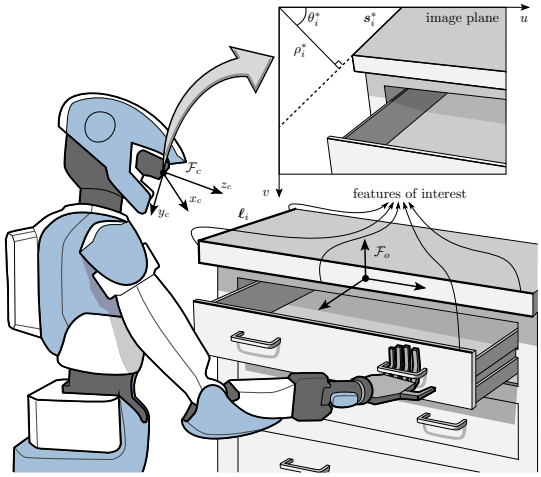


Fig. 1. A case example of domestic robotic manipulation: a humanoid pulls the drawer of a dresser. The scene observed by the robot camera is used to estimate the dresser configuration and achieve the manipulation task.

II. Background

Tracking of articulated systems, such as hands [3], human bodies [4] and robotic structures [5] is an active area of research in computer vision. However, these methods present serious drawbacks for feedback control: in general, they are (i) computationally demanding, (ii) not validated against ground-truth measurements (only superposed skeleton on image rendering), (iii) can experience convergence issues, (iv) some have jumps or jerkiness in frame-to-frame estimation, (v) some assume fixed cameras, and (vi) many are tailored for a specific item (e.g. only hands).

In order to be integrated into a robotic feedback control loop, a tracking algorithm has to be free from the mentioned drawbacks and rely on standard robot computational and sensory equipment. In this direction, joint encoder, depth

Manuscript received: November, 21, 2017; Revised February, 2, 2018; Accepted April, 23, 2018.

This paper was recommended for publication by Editor Jana Kosecka upon evaluation of the Associate Editor and Reviewers' comments. This work is supported in part by the H2020 COMANOID EU project and the CNRS-AIST-AIRBUS Joint Research Program.

All the authors are with the CNRS-University of Montpellier, Laboratoire d'Informatique de Robotique et de Microéectronique de Montpellier (LIRMM), IDH Group, France. author-surname@lirmm.fr

Digital Object Identifier (DOI): see top of this page.

images [6] or their combination [7], [8] can be used to track a robotic arm. However, these methods are unsuitable for objects not equipped with encoders, or would require that the robot is in stable contact with the object, which is restrictive.

RGB-D images are used to estimate pose, shape and structure of an object in [9]. In [10] a depth-based method, robust to calibration errors, estimates the state of a robotic arm in closed-loop manipulation tasks. However, for many manipulation tasks, the object-camera distance may not exceed the minimum range required by some depth sensors. Moreover, depth sensors using infrared do not work outdoor and those using LIDAR are computationally expensive. Stereo-vision, instead, needs additional steps to solve the reconstruction problem. This motivates us to use only monocular images.

Model-based approaches using single cameras are suitable to achieve our goal. The object geometric and appearance model are used in [11] to estimate the joints position and velocity, but not that of the floating base. In [12], the tracking of complex structures is handled imposing motion constraints to rigid objects tracking. Another interesting articulated object tracker has been proposed in [13]; it uses the virtual visual servoing (VVS) paradigm [14] and has real-time estimation capabilities [15]. Therefore, we took inspiration from [13] but our solution has a lighter formulation, is easier to implement, and estimates directly the object configuration. Moreover, we formulate the problem as a QP to be merged with QP-based control frameworks. In a previous work [16], we used points as visual features (structuring the object with known markers was required) and formulated the problem as a classic VVS.

Summing-up, the main features of our framework are:

- formulation of the articulated objects tracking as a QP;
- a perception algorithm providing the feedback based on an image processing procedure enhanced with the tracking information; structuring of the object is not required: line features, easily detectable at the object edges, are considered as feedback in the tracking;
- a learning-based procedure to automatically initialize our tracking;
- experiments assessing the effectiveness of our approach and the interlink between the estimation and the control.

III. Problem Formulation

We address the articulated objects estimation problem with the VVS paradigm. Visual servoing [17] provides the camera velocity $\boldsymbol{v}_c$ to achieve a cartesian task, zeroing the error between measured ($\boldsymbol{s}$) and desired ($\boldsymbol{s}^*$) visual features. Similarly, VVS computes the velocity of a *virtual* camera, whose unknown pose $\boldsymbol{p}$ in the cartesian space is defined in correspondence of some *virtual visual features* $\boldsymbol{s}(\boldsymbol{p})$, reconstructed by using the scene and the camera projection model. The real camera pose $\boldsymbol{p}^*$ is defined in correspondence of $\boldsymbol{s}^*$, the same visual features as measured on the image plane. The convergence of $\boldsymbol{s}(\boldsymbol{p})$ to $\boldsymbol{s}^*$ implies the convergence of $\boldsymbol{p}$ to $\boldsymbol{p}^*$. Thus, the VVS can estimate the camera pose integrating $\boldsymbol{v}_c$. This technique was introduced as an augmented reality tool [14], used to estimate the pose of an object moving in the camera frame, and extended to articulated objects in [13].

Fig. 1 shows a robot looking at the drawer of a dresser that needs to be opened to a given amount. The robot camera frame $\mathcal{F}_c$ is the reference having the focal point as origin with the z-axis aligned with the focal one. The image plane is defined by the u-v coordinates system, where the abscissa axis is oriented as x_c and the ordinate as y_c; $\mathcal{F}_o$ is a given floating base (FB) frame of the object.

The *real* configuration of the articulated object is

$$\boldsymbol{q}^* = \left(\boldsymbol{p}_o^{*T} \boldsymbol{\varphi}_o^{*T} \boldsymbol{q}_j^{*T}\right)^T \in \mathbb{R}^m, \tag{1}$$

where $\boldsymbol{p}_o^*$ and $\boldsymbol{\varphi}_o^*$ are the 3D position and orientation of $\mathcal{F}_o$ w.r.t. $\mathcal{F}_c$, respectively; $\boldsymbol{q}_j^* \in \mathbb{R}^n$ is the n joint coordinates vector of the articulated object.

The choice of the orientation parametrization is crucial, since $\mathcal{F}_o$ can change substantially w.r.t. $\mathcal{F}_c$. We choose a practical 3D orientation representation by applying the logarithmic map to the quaternion. This allows to use a minimal representation with the benefit of the unit quaternion (see more details in [18]). Using this representation, the FB pose can be expressed with 6 parameters (i.e., $m = 6 + n$).

Let $\boldsymbol{\ell}$ be a vector of f object features of interest (points, lines or any geometrical primitives of the object). To avoid any structuring of the object, we use lines, that are identified by three parameters in the cartesian space ($\boldsymbol{\ell} \in \mathbb{R}^{3f}$). The projection of $\boldsymbol{\ell}$ on the image plane provides f visual lines, each one expressed in the form $u \cos\theta_i + v \sin\theta_i = \rho_i$, where ρ_i is the length of the segment perpendicular to the i-th line and joining the origin of the image plane, and θ_i is the angle between this segment and the abscissa axis (see Fig. 1).

On the one side, the motion of the *real visual features* is induced from the articulated object-camera relative motion. Their parameters θ_i^* and ρ_i^* ($i = 1, \ldots, f$) are stacked in $\boldsymbol{s}^* \in \mathbb{R}^{2f}$. On the other side, the configuration $\boldsymbol{q}$ of the virtual articulated object, affects the motion of the *virtual visual features* $\theta_i(\boldsymbol{q})$ and $\rho_i(\boldsymbol{q})$, collected in $\boldsymbol{s}(\boldsymbol{q}) \in \mathbb{R}^{2f}$. They depend on: (i) the current estimation of $\boldsymbol{q}$, used to update the geometric model and calculate the lines $\boldsymbol{\ell}$ in $\mathcal{F}_c$ and, (ii) the camera intrinsic parameters, used to project the lines on the image plane. Since $\mathcal{F}_c$ is the reference, the camera extrinsic parameters are not needed for the projection.

The tracking algorithm estimates $\boldsymbol{q}^*$ given the object geometric model, containing N ordered lines $\boldsymbol{\ell}_m \in \mathbb{R}^{3N}$ ($\boldsymbol{\ell} \subseteq \boldsymbol{\ell}_m$) and vertices $\boldsymbol{p}_m \in \mathbb{R}^{3N}$ expressed in $\mathcal{F}_o$. The model is used to build $\boldsymbol{s}(\boldsymbol{q})$; $\boldsymbol{s}^*$ is measured on the image.

IV. Visual Tracking of Articulated Objects

In this section we describe our approach. Firstly, we recall the classical VVS and then propose the QP-based extension.

A. Classic VVS-based tracking scheme

The VVS error is the difference between virtual and real features, $\boldsymbol{e} = \boldsymbol{s}(\boldsymbol{q}) - \boldsymbol{s}^*$. Each virtual feature i motion obeys

$$\dot{\boldsymbol{s}}_i(\boldsymbol{q}) = -\boldsymbol{L}_i \boldsymbol{J}_i(\boldsymbol{q}) \dot{\boldsymbol{q}} = \boldsymbol{A}_i \dot{\boldsymbol{q}} \tag{2}$$

where the dot over the variables denotes the time derivative, $\boldsymbol{J}_i$ is the $6 \times m$ Jacobian of the object link where the i-th line

is defined (details are given in the Appendix), and $\boldsymbol{L}_i$ is the 2×6 image Jacobian associated to the i-th line [19]:

$$\boldsymbol{L}_i = \begin{pmatrix} \lambda_{\theta_i} c_{\theta_i} & \lambda_{\theta_i} c_{\theta_i} & -\lambda_{\theta_i}\rho & -\rho c_\theta & -\rho s_{\theta_i} & -1 \\ \lambda_{\rho_i} c_{\theta_i} & \lambda_{\rho_i} s_{\theta_i} & -\lambda_{\rho_i}\rho_i & s_{\theta_i}(1+{\rho_i}^2) & -c_{\theta_i}(1+\rho_i^2) & 0 \end{pmatrix} \tag{3}$$

with s_x being $\sin(x)$ and c_x being $\cos(x)$, $\lambda_{\theta_i} = (a_i s_{\theta_i} - b_i c_{\theta_i})/d_i$ and $\lambda_{\rho_i} = (a_i \rho_i s_{\theta_i} + b_i \rho_i c_{\theta_i} + c_i)/d_i$. The scalars a_i, b_i, c_i, d_i are the coefficients of a plane $\boldsymbol{\pi}_i$ supporting the i-th line, computed as $\boldsymbol{\pi}_i = \boldsymbol{P}_i \boldsymbol{p}_o$. $\boldsymbol{P}_i$ is the dual Plücker matrix associated to the line $\boldsymbol{\ell}_i$, and $\boldsymbol{p}_o$ is the estimated position of $\mathcal{F}_o$. Both $\boldsymbol{P}_i$ and $\boldsymbol{p}_o$ are available in the estimation routine as described below.

The dynamics of the error writes as $\dot{\boldsymbol{e}} = \boldsymbol{A}\dot{\boldsymbol{q}} - \dot{\boldsymbol{s}}^*$, where $\boldsymbol{A} = (\boldsymbol{A}_1, \ldots, \boldsymbol{A}_f)^T$ is the $2f \times m$ *articulation matrix* relating the FB and joints velocity to that of the visual features. Imposing a stable dynamics of the error, we have

$$\dot{\boldsymbol{q}} = -\lambda \boldsymbol{A}^{\#} \boldsymbol{e} + \boldsymbol{A}^{\#} \dot{\boldsymbol{s}}^* \tag{4}$$

where λ is a positive gain and $\boldsymbol{A}^{\#}$ the pseudoinverse of $\boldsymbol{A}$. At steady state, (4) gives an estimate of the object FB and joint velocities, from which $\boldsymbol{q}$ is obtained by integration.

B. Formulation of the VVS-based tracking as a QP

We formulate the estimation problem as a QP for the following reasons: it allows to (i) take into account modelling and measurement error [20] (ii) add constraints (equalities or inequalities) on the estimation variables (e.g. joint limits), and more importantly (iii) integrate the estimation to the control in the same unified framework. Indeed, since many robotic control framework are based on a QP that also includes visual servoing tasks [21], it is reasonable to think about the articulated object tracker written as a QP.

The VVS rationale in Sect. IV-A can be written as a minimization problem. Since the decision variable of the QP robot controller (see Sect. VII) is the acceleration of the multi-robot configuration, it is convenient to define $\ddot{\boldsymbol{q}}$ as state of our "QP-VVS". Thus, the objective function writes:

$$\boldsymbol{f}_o(\ddot{\boldsymbol{q}}) = \tfrac{1}{2} \| k(\boldsymbol{s}^* - \boldsymbol{s}(\boldsymbol{q})) + b(\dot{\boldsymbol{s}}^* - \dot{\boldsymbol{s}}(\boldsymbol{q})) + (\ddot{\boldsymbol{s}}^* - \ddot{\boldsymbol{s}}(\boldsymbol{q})) \|^2 \tag{5}$$

with k a positive constant gain and $b = 2\sqrt{k}$. From (2), it is

$$\ddot{\boldsymbol{s}}_i(\boldsymbol{q}) = \dot{\boldsymbol{A}}_i \dot{\boldsymbol{q}} + \boldsymbol{A}_i \ddot{\boldsymbol{q}} \tag{6}$$

where $\dot{\boldsymbol{A}}_i = -\dot{\boldsymbol{L}}_i \boldsymbol{J}_i - \boldsymbol{L}_i \dot{\boldsymbol{J}}_i$, $i = 1, \ldots, f$. Substituting (2) and (6) in (5), the QP providing the double derivative of the articulated object configuration can be written as follows:

$$\begin{aligned} \ddot{\boldsymbol{q}} &= \underset{\ddot{\boldsymbol{q}} \in \mathcal{S}}{\arg\min}\ \tfrac{1}{2} \ddot{\boldsymbol{q}}^T \boldsymbol{Q} \ddot{\boldsymbol{q}} + \boldsymbol{c}^T \ddot{\boldsymbol{q}} \\ \mathcal{S} &= \{ \ddot{\boldsymbol{q}} \mid \boldsymbol{K} \ddot{\boldsymbol{q}} \leq \boldsymbol{k} \} \end{aligned} \tag{7}$$

from which $\boldsymbol{q}$ is obtained by numerical integration. In (7), $\boldsymbol{Q} = \boldsymbol{A}^T \boldsymbol{A}$, $\boldsymbol{c} = -\boldsymbol{A}^T [k(\boldsymbol{s}^* - \boldsymbol{s}(\boldsymbol{q})) + b(\dot{\boldsymbol{s}}^* - \boldsymbol{A}\dot{\boldsymbol{q}}) + \ddot{\boldsymbol{s}}^* - \dot{\boldsymbol{A}}\dot{\boldsymbol{q}}]$; $\ddot{\boldsymbol{s}}^*$ and $\dot{\boldsymbol{s}}^*$ are obtained by numerical derivation. $\mathcal{S}$ represents the set of feasible $\ddot{\boldsymbol{q}}$ accounting for the kinematic and dynamic constraints of the articulated object. For instance, to impose limits on $\boldsymbol{q}$, one could set $\boldsymbol{K} = [-\boldsymbol{I}_m, \boldsymbol{I}_m]^T$ and $\boldsymbol{k} = [-2(\boldsymbol{q}_{\min}^T - \boldsymbol{q}^T - \dot{\boldsymbol{q}}^T T_s)/T_s^2, 2(\boldsymbol{q}_{\max}^T - \boldsymbol{q}^T - \dot{\boldsymbol{q}}^T T_s)/T_s^2]^T$ [2], where $\boldsymbol{I}_m$ is the $m \times m$ identity matrix; $\boldsymbol{q}_{\min}$ and $\boldsymbol{q}_{\max}$ define the range in which $\boldsymbol{q}$ is constrained; T_s is the sampling time.

for each new image frame $\boldsymbol{I}$ **do**
  $\boldsymbol{s}^* \leftarrow$ TRACK FEATURES($\boldsymbol{I}$)
  while $||\boldsymbol{e}_\theta|| > e_{\text{th},\theta} \wedge ||\boldsymbol{e}_\rho|| > e_{\text{th},\rho}$ **do**
    $\boldsymbol{\ell} \leftarrow$ UPDATE MODEL($\boldsymbol{q}$)
    for each visual feature i **do**
      $\boldsymbol{J}_i \leftarrow$ COMPUTE JACOBIAN($\boldsymbol{q}$)
      $\dot{\boldsymbol{J}}_i \leftarrow$ COMPUTE JAC. DERIVATIVE($\boldsymbol{q}, \dot{\boldsymbol{q}}$)
      $\boldsymbol{\pi}_i \leftarrow$ COMPUTE SUPPORTING PLANE($\boldsymbol{\ell}_i, \boldsymbol{q}$)
      $\boldsymbol{s}_i(\boldsymbol{q}) \leftarrow$ PROJECT($\boldsymbol{\ell}_i$)
      $\boldsymbol{L}_i \leftarrow$ COMPUTE IMAGE JACOBIAN($\boldsymbol{\pi}_i, \boldsymbol{s}_i(\boldsymbol{q})$)
      $\dot{\boldsymbol{L}}_i \leftarrow$ COMPUTE IM. JAC. DER.($\boldsymbol{\pi}_i, \boldsymbol{s}_i, \dot{\boldsymbol{\pi}}_i, \dot{\boldsymbol{s}}_i$)
      $\boldsymbol{A}_i = -\boldsymbol{L}_i \boldsymbol{J}_i$, $\dot{\boldsymbol{A}}_i = -\dot{\boldsymbol{L}}_i \boldsymbol{J}_i - \boldsymbol{L}_i \dot{\boldsymbol{J}}_i$
    end for
    $\ddot{\boldsymbol{q}} = \arg\min \frac{1}{2} \ddot{\boldsymbol{q}}^T \boldsymbol{Q} \ddot{\boldsymbol{q}} + \boldsymbol{c}^T \ddot{\boldsymbol{q}}$
    $\boldsymbol{q} \leftarrow$ DOUBLE INTEGRATION($\ddot{\boldsymbol{q}}$)
  end while
end for

Fig. 2. Algorithm realizing the VVS-based tracking of articulated objects.

Fig. 2 presents the whole routine computing $\boldsymbol{q}$. An edge tracker (see Sect. V-C) processes the image $\boldsymbol{I}$ acquired by the camera to provide the vector $\boldsymbol{s}^*$. Then, using the current estimation $\boldsymbol{q}$, the model of the object is updated, so that the positions of the features $\boldsymbol{\ell}$ are also estimated and available for subsequent computations. Thus, for each visual features $i = 1, \ldots, f$ the following actions are carried out:

- Jacobian $\boldsymbol{J}_i$ and its time derivative $\dot{\boldsymbol{J}}_i$ are computed using current $\boldsymbol{q}$ and $\dot{\boldsymbol{q}}$ (see the Appendix);
- a supporting plane $\boldsymbol{\pi}_i$ is computed using the line Plücker coordinates and the current estimate of the FB position; the derivative $\dot{\boldsymbol{\pi}}_i$ is also numerically computed;
- virtual visual feature $\boldsymbol{s}_i(\boldsymbol{q})$ is obtained projecting $\boldsymbol{\ell}_i$ on the image plane, using the camera projection model;
- the image Jacobian $\boldsymbol{L}_i$ and its time derivative $\dot{\boldsymbol{L}}_i$ are computed using the estimate of the plane and the coordinates of the visual feature, and their time derivatives;
- the i-th block of the articulation matrix, $\boldsymbol{A}_i$, is obtained.

At the end of these steps, the articulation matrix is fully built, $\boldsymbol{Q}$ and $\boldsymbol{c}$ can be computed and the QP solved. Since the estimate is accurate only when the tracking converges, an inner loop repeats the tracking operations until the norm of the VVS error on θ and ρ ($\boldsymbol{e}_\theta$ and $\boldsymbol{e}_\rho$, respectively) decreases below desired precision thresholds $e_{\text{th},\theta}$ and $e_{\text{th},\rho}$.

V. PERCEPTION OF THE VISUAL FEATURES

We explain the methods used to measure $\boldsymbol{s}^*$ and reconstruct $\boldsymbol{s}(\boldsymbol{q})$, and how the perception algorithm copes with the initialization phase, the missing/occluded lines and the real-virtual features correspondence. The whole algorithm is based on the so-called `edges_table` that contains the following data for each of the N edges of the object model:

- `id`, its identification number;
- `part_id`, the identification number of its object link;
- `tracked`, its boolean indicating if tracked;

- `visible`, boolean indicating its visiblity on the image;
- its θ and ρ visual line parameters on the image plane;
- $\bar{\theta}$ and $\underline{\theta}$ ($\bar{\rho}$ and $\underline{\rho}$) upper and lower bounds of its θ (ρ), respectively; they define a range where to search for it;
- $\boldsymbol{p}_1$ and $\boldsymbol{p}_2$, its extreme points used to define a region of interest (ROI) where to search for it;
- l, its length.

The geometric information (θ-ρ parameters, $\boldsymbol{p}_1$, $\boldsymbol{p}_2$, l) are given only for the tracked edges, i.e., $\forall\ \mathtt{id} \in \mathcal{T} = \{1, \ldots, N \mid \mathtt{tracked}_{\mathtt{id}} = \text{“true”}\}$. The `part_id` field is known in advance and read from the object geometric model. All the other data is updated at each new acquired image by the perception algorithm, as described below.

A. Initialization

To make the algorithm start, the `edges_table` and the configuration vector $\boldsymbol{q}$ have to be initialized. To this end, two initialization procedures are provided: manual and automatic.

Firstly, a set of f edges to track is decided (see Sec. VI), to which the corresponding `tracked` value is set to “true”. For all the other edges, this field is initialized to “false”.

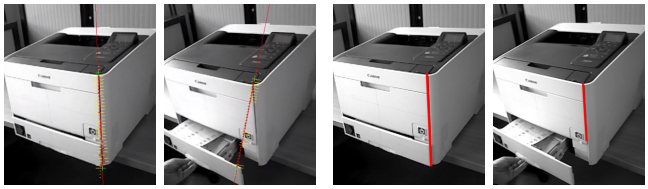
The manual initialization consists in pointing on the image the f chosen edges one by one. This operation fills the corresponding `edges_table` fields related to the θ-ρ parameters, $\boldsymbol{p}_{1,2}$ and, consequently, l. The upper and lower bounds are initialized as $\underline{\theta}_{\mathtt{id}} = \theta_{\mathtt{id}} - m_\theta$, $\bar{\theta}_{\mathtt{id}} = \theta_{\mathtt{id}} + m_\theta$, $\underline{\rho}_{\mathtt{id}} = \rho_{\mathtt{id}} - m_\rho$ and $\bar{\rho}_{\mathtt{id}} = \rho_{\mathtt{id}} + m_\rho$, $\forall\ \mathtt{id} \in \mathcal{T}$; m_θ and m_ρ are two heuristic margins, used to be conservative with the lines searching operations. Furthemore, selecting 4 points on the image of the object, and considering the corresponding points on the object in $\mathcal{F}_o$ (stored in the model), the initial estimate $\boldsymbol{q}$ is obtained by solving a perspective-n-point problem.

The automatic initialization is enabled by learning an object appearance model. To create a dataset for the learning, one visual marker is placed on the object, and image frames containing the object are acquired. The frames are processed to remove the visual marker by replacing area behind it with nearby pixel values, while using the marker pose to project predefined set of object landmarks onto the image. The pose of the marker is not always well estimated. Thus, a round of manual check is required to ensure high quality of the training data. The resulting dataset is used to learn the histogram of oriented gradients (HOG) filter [22] and to train landmark prediction model [23]. During the initialization phase, learned models are applied to the first frame(s), 4 detected landmarks on the FB are used in order to estimate $\boldsymbol{q}$, which is then used to project auxiliary landmarks to the image for estimation of initial tracked lines extreme points and, consequently, the θ-ρ parameters and length.

B. Reconstruction of the Virtual Line Features

The reconstruction of the virtual lines consists in filling the vector $\boldsymbol{s}(\boldsymbol{q})$, given the current value of the object configuration $\boldsymbol{q}$. During this process, the model information is used to update `visible`, $\boldsymbol{p}_1$ and $\boldsymbol{p}_2$ in the `edges_table`.

The virtual lines are obtained by executing three steps. First, the current estimate of the object configuration $\boldsymbol{q}$ is used to update the model, i.e., compute the model lines $\boldsymbol{\ell}_m$ in $\mathcal{F}_c$. Then, the `tracked` field of the table is used to select, among all the $\boldsymbol{\ell}_m$, the f features $\boldsymbol{\ell}$ to be considered for the VVS. The correspondence between the virtual and real features is made at this stage, since the `tracked` field is also used to perform the detection of the real lines (see Sect. V-C). Then, the projection of $\boldsymbol{\ell}$ on the image plane gives the vector $\boldsymbol{s}(\boldsymbol{q})$. For all the lines whose `tracked` value is false, $\boldsymbol{p}_1$ and $\boldsymbol{p}_2$ are updated with the edges vertices contained in the model ($\boldsymbol{p}_m$). Furthermore, using the information of the lines $\boldsymbol{\ell}_m$ in $\mathcal{F}_c$, it is possible to find out the lines that are visible, and update the `visible` field accordingly.



(a) *Moving-edge* used in [13]. (b) *Hough-guided* used in this work.

Fig. 3. Comparison between two algorithms to track the real line features.

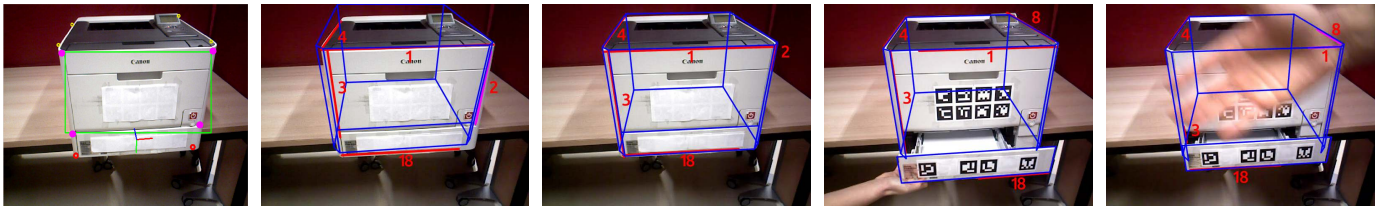
C. Tracking of the Real Line Features

The tracking of the real lines is based on the processing of the current image $\boldsymbol{I}$ acquired by the camera and, using the current information contained in the `edges_table`, filling the vector $\boldsymbol{s}^*$ and updating some of the table entries.

In [13] and [14] the lines at the borders of an object are tracked using the so-called *moving-edge* algorithm. We encountered problems to apply this technique when the parts of the articulated object move apart (see Fig. 3a). Therefore, we developed a *Hough-guided* procedure consisting of the following steps. Firstly, apply the Canny operator to provide a black and white map of the edges in the image. For each tracked line at the previous image (`tracked` is true), select a ROI and then apply on it the Hough transform of the edges map to detect the edges extreme points from which the θ-ρ are computed (a typical result is shown in Fig. 3b).

For each edge $\mathtt{id} \in \mathcal{T}$, the ROI selection is performed according to the values stored in $\boldsymbol{p}_1$ and $\boldsymbol{p}_2$ and using two margins $m_{\text{ROI},x}$ and $m_{\text{ROI},y}$. Furthermore, to refine the searching of the line, only the Hough's output ranging in $(\underline{\theta}, \bar{\theta})$ and $(\underline{\rho}, \bar{\rho})$ are considered and merged together.

If the detection process succeeds in finding the edge, then the θ-ρ parameters given by Hough are used to update the table and compose the vector $\boldsymbol{s}^*$. The extreme points are used to fill the $\boldsymbol{p}_1$ and $\boldsymbol{p}_2$ fields. If the detection process fails, the `tracked` entry for that line is set to “false”. In this case, a recovery strategy is activated to find a substitute among the lines in the table that have the `part_id` same as the line just lost, and whose `visible` value is “true”. If the new detected line has a length (l) greater than a threshold, then it is considered in the tracking. Otherwise, the last detected edge is kept in the table. The recovery strategy also helps the tracker with temporary failures of the detection, trying to regain the convergence to the real object from the current status of the `edges_table`. The θ-ρ



(a) (b) (c) (d) (e)

Fig. 4. First tracking experiment. The learning-based algorithm recognizes the parts of the printer (a) and provides VVS with initialization: the green rectangle shows HOG-based detection, the purple dots show prediction of the front face landmarks that are used to compute the initial pose estimate (displayed as a RGB frame); yellow and red points are auxiliary landmarks projected on the image using the estimate $\boldsymbol{q}$. After a transient phase (b), the virtual features (blue lines) converge to their real red counterparts (c), and correctly track the printer FB and the paper tray motion (d). The algorithm copes with visual occlusions (e), and accidental failures of the lines detection: e.g., in (d) line 2 is substituted with line 8.

bounds are updated as $\bar{x}_{\mathtt{id}} = \max\left(x_{\mathtt{id}}, x_{\mathtt{id}} + \dot{x}_{\mathtt{id}} T_s\right) + m_x$, and $\underline{x}_{\mathtt{id}} = \min\left(x_{\mathtt{id}}, x_{\mathtt{id}} + \dot{x}_{\mathtt{id}} T_s\right) - m_x$, $\forall\ \mathtt{id} \in \mathcal{T}$ and $x \in \{\theta, \rho\}$. In the above relationships, the derivative term can be computed numerically or, if the tracker converges, using the information available in the VVS, i.e., $(\dot{\theta}_{\mathtt{id}}, \dot{\rho}_{\mathtt{id}})^T \approx \boldsymbol{A}_i \dot{\boldsymbol{q}}$.

VI. Experimental Results

The presented results are obtained by tracking two articulated objects: a printer with the prismatic joint of its paper tray, and a cabinet with its door revolute joint. The tracking provides the objects FB pose and the joint variable ($m = 7$).

Our results are obtained using different numbers of line features. Indeed, the VVS estimation depends on the number of the f tracked lines and their relative position. For example, to correctly estimate the FB pose, a minimal set of 3 non-coplanar lines is required to ensure that the stacked image Jacobian, and consequently the articulation matrix, is full rank (the same problem is remarked in the dual context of VS [19]). The lines coplanarity can be detected with additional computation. However, we decided to track more lines than the minimal required, to create redundancy and avoid rank deficiency. This solution, paid in terms of a higher computational cost, is shown to be feasible in the results presented below. Furthermore, increasing the dimension of the feedback helps to have more robust tracking results with respect to the perception process noise and bad detection of the line features. Thus, we chose the lines number f driven by the trade-off between high tracking performance and low computational cost. Note that the algorithm (shown in Fig. 2) scales linearly with f. In the experiments with the printer we used 4 lines on the FB and one on the paper tray; for the cabinet we used 5 lines on the FB and 2 on the door.

The algorithm run on a PC with an i7 2.60 GHz CPU and 8 GB of RAM, processing the images from the monocular camera of a XtionPRO live RGB-D sensor, that gives a video stream of 640×480 pixels at 30 Hz. Since the frequency of the estimation process is, in general, higher, the remaining time is used by the inner loop to make the VVS converge. An offline calibration procedure provides the camera intrinsic parameters used to compute the projection model. The real lines detection algorithm is based on the OpenCV library. A low-pass frequency filter is used to clear the noise from the configuration estimate, the detected lines and their derivative.

We propose two sets of experiments. The first aims at evaluating the tracking performance, the latter shows the effectiveness of the tracking for robotic manipulation tasks. All the experiments, along with the online camera images, are included in the video accompanying this paper.

A. Visual Tracking of Articulated Objects

As detailed in Sect. IV, we used as feedback only the lines that are detectable at the borders of the objects. However, for the first set of experiments, we also placed two markers boards on the FB and on the link of the objects. These markers are not considered in our estimator but processed by the Aruco library [24] to reconstruct the objects configuration, considered for comparison in our work. For this set of experiments, the thresholds of the error used to stop the inner loop were tuned to $e_{\text{th},\theta} = e_{\text{th},\rho} = 0.015$.

In the first experiment we tracked the printer, initializing the algorithm with the automatic procedure (Sect. V-A). The parameters for the lines searching algorithm (Sect. V-C) were set to $m_\theta = 0.035$ and $m_\rho = 12$ whereas the margins on the ROI were set to $m_{\text{ROI},x} = 30$ and $m_{\text{ROI},y} = 10$. The VVS-QP gain was heuristically set to $k = 750$. In this experiment, the average time spent by the lines tracking process was 5.79 ms, and 8.31 ms by the VVS estimation (corresponding to an average of 366 iterations of the inner loop). The tracking process steps and performance are shown in Fig. 4.

In the second experiment, we tracked the cabinet. We intentionally bad initialized the algorithm using the manual procedure. The parameters of the algorithm were set as follows: $m_\theta = 0.035$, $m_\rho = 10$, $m_{\text{ROI},x} = 45$ and $m_{\text{ROI},y} = 25$; $k = 2500$. On average, the lines tracking took 5.63 ms, the estimation 12.83 ms (435 inner iterations). During the experiment the camera moves w.r.t. the object and *vice-versa*. Figures 5 and 6 show the plots of the cabinet position and orientation (transformed in roll-pitch-yaw angles), respectively. After an initial transient time required to recover the bad initialization, the signals provided by the VVS (blue traces with triangular markers) converge to the pose of the cabinet. The results are compared with the Aruco signals (red dashed lines). The door joint angle is plotted in Fig. 7: the VVS and Aruco output match quite well along all the experiment.

In a third experiment, the printer is tracked in a cluttered environment (Fig. 8a) using both the classic and QP formalism of the VVS. Details of the tracker show that the QP-VVS benefits from the inclusion of the kinematic constraints (Fig. 8b)

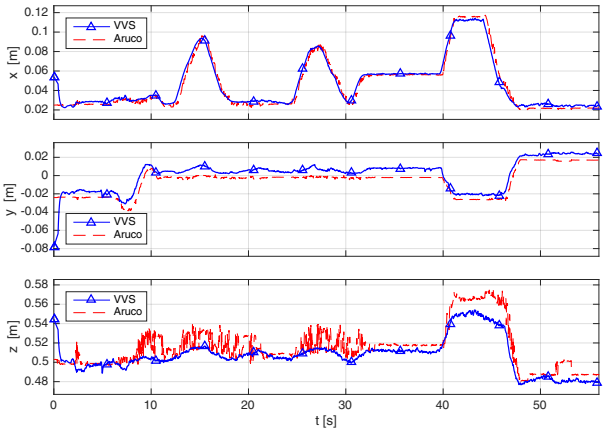


Fig. 5. Second tracking experiment: the cabinet FB position.

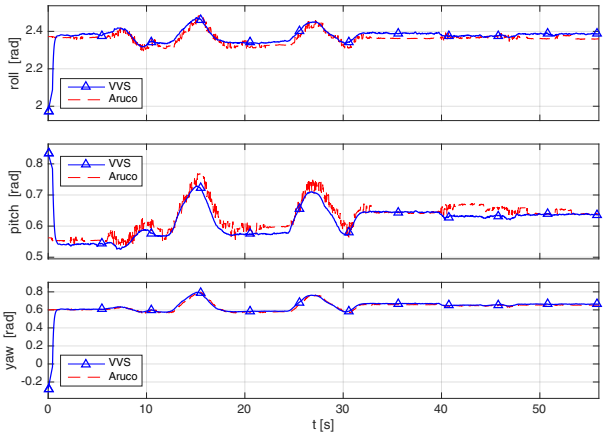


Fig. 6. Second tracking experiment: the cabinet FB orientation.

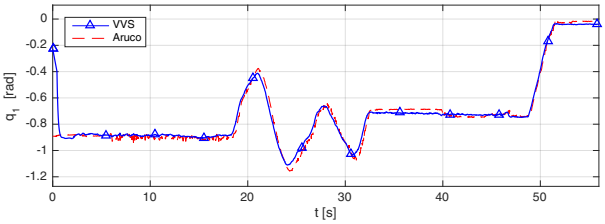


Fig. 7. Second tracking experiment: the cabinet revolute joint angle.

when the noise on the detected lines can produce results not consistent with the object geometry (Fig. 8c).

B. Robotic Manipulation of Articulated Objects

In the second set of experiments we made HRP-4 robot manipulate the articulated objects. To this end, the estimation output given by the VVS-based scheme is used as feedback in the MQP framework to control the "robot+object" system, as mentioned in Sect. I. Among the others, the framework allows to define *contact* constraints between the robot and the object to be manipulated, actually coupling the two parts in a single system. Once that the contact is established, the overall system can be controlled with both *cartesian* and *postural* tasks. Indeed, in order to grasp the object, a cartesian error $\boldsymbol{\tau}_h = \boldsymbol{p}_h - \boldsymbol{p}_{h,d}$ is defined, where $\boldsymbol{p}_h$ is the current value of

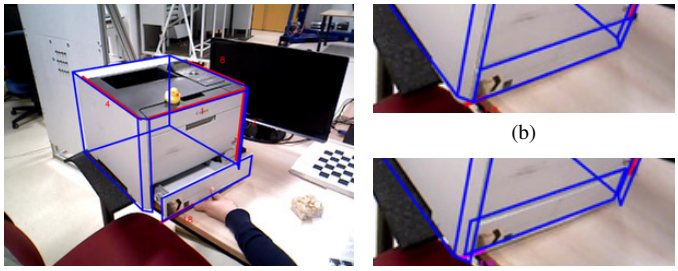


Fig. 8. Third tracking experiment: the printer in a cluttered evironment (a). Detail of the tracker using the the QP-VVS (b) and the classic formalism (c).

the robot hand pose and $\boldsymbol{p}_{h,d}$ is the object pose to be grasped, given by the tracking algorithm. For the manipulation task, an error is defined as $\tau_q = q_1 - q_{1,d}$, where $q_{1,d}$ (desired object joint value) is provided by some form of planning, while q_1 (current object joint value) is estimated by our method. To execute the grasping or the manipulation task, a new term is properly added in the cost function of the MQP, that tries to zero the corresponding error. Each term is given a gain, imposing a decrease rate of the task error, and a weight, defining the priority of the task. The constraints and the tasks are added or removed in the MQP with a state machine.

In this set of experiments, we used the manual initialization. We tuned the parameters of the perception algorithm as follows: $m_\theta = 0.35$, m_ρ equal to 12 for the printer and 10 for the cabinet, $m_{\text{ROI},x} = 40$ and $m_{\text{ROI},y} = 10$. The gain of the VVS-QP was tuned to $k = 750$ for the printer and $k = 1250$ for the cabinet; finally, we set to $e_{\text{th},\theta} = e_{\text{th},\rho} = 0.005$.

In the first manipulation experiment, HRP-4 operates the paper tray of the printer. The robot starts from its operational configuration, standing in front of the printer and grasping the tray. Several open/close commands are then sent to the MQP. The plot of Fig. 9 shows the manipulation commands (black dash-dot line) that are well followed by the printer prismatic joint, as estimated by the VVS method (blue line with triangular markers). With reference to the plot, the contact constraint between the robot hand and the printer is activated around time 8 s, after which the robot moves the tray toward the default desired position, that is 0 m; the first opening motion is sent at 12 s. One can observe a lag between command and tracking. This is due to the backlash in the humanoid-printer system, and the MQP task error decrease rate. One could also observe a not perfect positioning of the tray (e.g., at 38 s or 72 s). This is due to the high friction of the tray mechanism, preventing a smooth manipulation motion

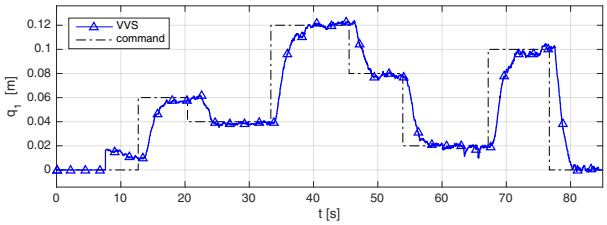
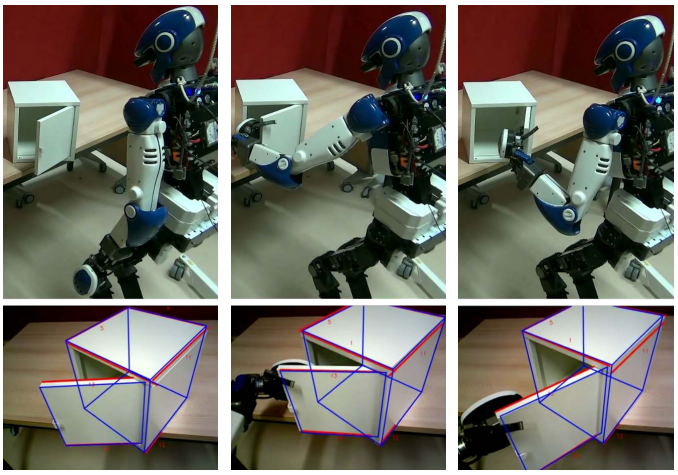


Fig. 9. First manipulation experiment: the printer prismatic joint position.



(a) (b) (c)

Fig. 10. Second manipulation experiment. HRP-4 is arbitrarily placed in front of a cabinet (a). The VVS estimate is used to steer the robot hand to the door, grasp (b) and open it (c) by adding MQP cartesian/postural tasks.

and a fine positioning. In this experiment, the lines and object tracking process took an average of 9.79 ms and 8.8 ms (409 inner iterations).

In the second manipulation experiment we achieve both grasping and operation of the cabinet door. Three significative snapshots taken from the experiment video are shown in Fig. 10. Fig. 11 shows that the angle of the door remains constant while the robot hand is reaching the object. At time 22 s, the hand touches the door and accidentally closes it by 0.1 rad. Then, at 25 s, the contact constraint is activated and the robot steer the door at the default command (0.8 rad). Finally, the commands are sent at about 29 s and HRP-4 performs the opening/closing motion as specified by the user. On an average, the tracking of the lines used 10.38 ms and the estimation process 8.71 ms (for 277 inner loop iterations).

VII. Conclusion and Future Work

Our online articulated objects estimator can be written as a virtual visual servoing quadratic program tracker in the acceleration space. The estimator and the control framework cooperates to achieve the desired manipulation task, sharing information (such as the object model). Interesting perspectives are currently under investigation. In fact, the estimation can be viewed as an observer that can be explicitly merged with the whole body robot controller in a more effective single MQP formulation. In doing so, the idea of *task* in the robot control MQP formulation could be generalized to include also the meaning of estimation. The fundamental challenge is to find the appropriate conditions to make the "observer" tasks part of the MQP converge faster then the control *per se* tasks, since the latter depends on the first.

We have also shown that our algorithm is able to track non-structured objects, cope with missing/occluded features and cluttered environments. However, the use of line features, as well as any other geometric features, represents also a limitation. First, not all objects can be identified with simple geometric features. Second, this kind of information is prone

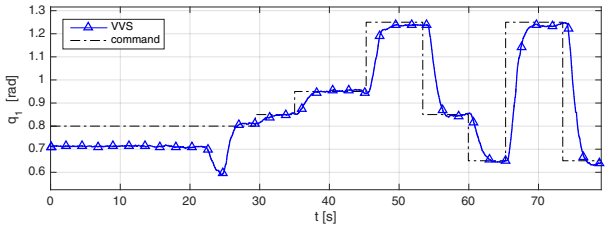


Fig. 11. Second manipulation experiment: the cabinet revolute joint angle.

to be mistaken with similar visible features not related to the object. Future work will investigate the possibility to include in the tracker also other kind of sensory information, such as reconstructed depth (under certain working conditions), robot joints encoders (when the robot is in contact with the object) and other features that can be learned robustly offline.

Tracking and manipulation experiments carried out with HRP-4 have shown the effectiveness of our approach to be used in closed-loop control. We believe that our algorithm can be a good basis for an extension to the tracking of more complex structures such as human bodies and employed in the field of physical human-robot interaction.

Appendix

The tracking algorithm described in Sect. IV uses the articulation matrix and its time derivative, i.e., the object and image Jacobian related to the i-th line feature ($\boldsymbol{J}_i$ and $\boldsymbol{L}_i$, respectively) and their derivatives ($\dot{\boldsymbol{J}}_i$ and $\dot{\boldsymbol{L}}_i$). The first step is to write the FB Jacobian, $\boldsymbol{J}_o$. Recall that the FB orientation $\boldsymbol{\varphi}_o = (\varphi_1, \varphi_2, \varphi_3)^T$ is obtained applying a *log* map [18] to the unit quaternion $\boldsymbol{\sigma}_o = (\eta, \boldsymbol{\varepsilon}^T)^T$, $\boldsymbol{\varepsilon} = (\varepsilon_1, \varepsilon_2, \varepsilon_3)^T$:

$$\boldsymbol{\varphi}_o = \log \boldsymbol{\sigma}_o = \frac{2\cos^{-1}\eta}{|\boldsymbol{\varepsilon}|}\boldsymbol{\varepsilon}. \tag{8}$$

From $\boldsymbol{\varphi}_o$, the rotation matrix $\boldsymbol{R}_o$ from $\mathcal{F}_c$ to $\mathcal{F}_o$ is obtained applying the *exp* map to express the orientation in quaternion:

$$\boldsymbol{\sigma}_o = \exp \boldsymbol{\varphi}_o = \left(\cos\bar{\alpha}, \frac{\sin\bar{\alpha}}{\alpha}\boldsymbol{\varphi}_o\right)^T \tag{9}$$

where $\alpha = |\boldsymbol{\varphi}_o|$ and $\bar{\alpha} = \alpha/2$; then, $\boldsymbol{R}_o$ is extracted from the quaternion using the well known relation. The FB angular velocity is related to the orientation parameters derivative:

$$\boldsymbol{\omega}_o = 2\boldsymbol{E}^T\dot{\boldsymbol{\sigma}}_o = 2\boldsymbol{E}^T\boldsymbol{J}_\varphi\dot{\boldsymbol{\varphi}}_o. \tag{10}$$

$\boldsymbol{E}$ is derived from the quaternion propagation rule:

$$\boldsymbol{E} = \begin{pmatrix} -\boldsymbol{\varepsilon}^T \\ \eta\boldsymbol{I}_3 - \boldsymbol{S}(\boldsymbol{\varepsilon}) \end{pmatrix} \tag{11}$$

where $\boldsymbol{I}_3$ is the 3×3 identity matrix and $\boldsymbol{S}(\boldsymbol{\varepsilon})$ the skew symmetric matrix associated to $\boldsymbol{\varepsilon}$. In (10), $\boldsymbol{J}_\varphi$ is the Jacobian of $\boldsymbol{\sigma}_o$ w.r.t. $\boldsymbol{\varphi}_o$, given in [18] and here reported element-wise:

$$J_\varphi^{1,j} = \frac{\partial\eta}{\partial\varphi_j} = -\varphi_j\frac{s_{\bar{\alpha}}}{2\alpha}, \tag{14}$$

$$J_\varphi^{i+1,j} = \frac{\partial\varepsilon_i}{\partial\varphi_j} = \begin{cases} \varphi_j^2\left(\frac{c_{\bar{\alpha}}}{2\alpha^2} - \frac{s_{\bar{\alpha}}}{\alpha^3}\right) + \frac{s_{\bar{\alpha}}}{\alpha} & \text{if } i = j, \\ \varphi_i\varphi_j\left(\frac{c_{\bar{\alpha}}}{2\alpha^2} - \frac{s_{\bar{\alpha}}}{\alpha^3}\right) & \text{if } i \neq j, \end{cases} \tag{15}$$

whereas, the derivative of $\boldsymbol{J}_\varphi$ is here expressed element-wise

$$\dot{J}_\varphi^{1,j} = -\frac{1}{2\alpha}\left(\dot{\varphi}_j s_{\bar{\alpha}} + \varphi_j\dot{\alpha}\frac{c_{\bar{\alpha}}\alpha - 2s_{\bar{\alpha}}}{2\alpha}\right), \tag{16}$$

$$\dot{\boldsymbol{L}}_i = \begin{pmatrix} \dot{\lambda}_{\theta_i} s_{\theta_i} + \lambda_{\theta_i} c_{\theta_i} \dot{\theta}_i & \dot{\lambda}_{\theta_i} c_{\theta_i} - \lambda_{\theta_i} s_{\theta_i} \dot{\theta}_i & -\dot{\lambda}_{\theta_i} \rho_i - \lambda_{\theta_i} \dot{\rho}_i & \dot{\rho}_i s_{\theta_i} + \rho_i c_{\theta_i} \dot{\theta}_i & \dot{\rho}_i c_{\theta_i} - \rho_i s_{\theta_i} \dot{\theta}_i & 0 \\ \dot{\lambda}_{\rho_i} s_{\theta_i} + \lambda_{\rho_i} c_{\theta_i} \dot{\theta}_i & \dot{\lambda}_{\rho_i} c_{\theta_i} - \lambda_{\rho_i} s_{\theta_i} \dot{\theta}_i & -\dot{\lambda}_{\rho_i} \rho_i - \lambda_{\rho_i} \dot{\rho}_i & -s_{\theta_i} \dot{\theta} (1+\rho_i^2) + 2 c_{\theta_i} \rho_i \dot{\rho}_i & -c_{\theta_i} \dot{\theta}_i (1+\rho_i^2) - 2 s_{\theta_i} \rho_i \dot{\rho}_i & 0 \end{pmatrix} \tag{24}$$

$$\dot{\lambda}_{\theta_i} = \left[(-\dot{a}_i c_{\theta_i} + a s_{\theta_i} \dot{\theta}_i + \dot{b} s_{\theta_i} + b c_{\theta_i} \dot{\theta}_i) d_i + (a_i c_{\theta_i} - b s_{\theta_i}) \dot{d}_i \right] / d_i^2 \tag{25}$$

$$\dot{\lambda}_{\rho_i} = \left[(\dot{a}_i \rho_i s_{\theta_i} + a_i \dot{\rho}_i s_{\theta_i} + a_i \rho_i c_{\theta_i} \dot{\theta}_i + \dot{b}_i \rho_i c_{\theta_i} + b_i \dot{\rho}_i c_{\theta_i} - b_i \rho_i s_{\theta_i} \dot{\theta}_i + c_i) d_i - (a_i \rho_i s_{\theta_i} + b_i \rho_i c_{\theta_i} + c_i) \dot{d}_i \right] / d_i^2 \tag{26}$$

$$\begin{aligned} \dot{J}_{\varphi}^{i+1,j} = & \tfrac{\varphi_j \dot{\varphi}_j}{\alpha^2} \left(c_{\bar{\alpha}} - \tfrac{s_{\bar{\alpha}}}{\bar{\alpha}} \right) + \tfrac{\dot{\alpha}}{\alpha} \left(\tfrac{c_{\bar{\alpha}}}{2} - \tfrac{s_{\bar{\alpha}}}{\alpha} \right) \\ & + \tfrac{\varphi_j^2 \dot{\alpha}}{\alpha^2} \left(s_{\bar{\alpha}} \tfrac{12-\alpha^2}{4\alpha^2} - \tfrac{3 c_{\bar{\alpha}}}{2\alpha} \right) \text{if } i = j, \end{aligned} \tag{17}$$

$$\begin{aligned} \dot{J}_{\varphi}^{i+1,j} = & \tfrac{\varphi_i \dot{\varphi}_j + \dot{\varphi}_i \varphi_j}{\alpha^2} \left(\tfrac{c_{\bar{\alpha}}}{2} - \tfrac{s_{\bar{\alpha}}}{\bar{\alpha}} \right) \\ & + \tfrac{\varphi_i \varphi_j \dot{\alpha}}{\alpha^2} \left(s_{\bar{\alpha}} \tfrac{12-\alpha^2}{4\alpha^2} - \tfrac{3 c_{\bar{\alpha}}}{2\alpha} \right) \text{if } i \neq j, \end{aligned} \tag{18}$$

with $\dot{\alpha} = (\boldsymbol{\varphi} \cdot \dot{\boldsymbol{\varphi}})/\alpha$. Approximations, using Taylor expansion of sine and cosine, are considered for $\alpha \simeq 0$ [18]. From (10):

$$\boldsymbol{J}_o = \begin{pmatrix} \boldsymbol{I}_3 & \boldsymbol{O}_3 \\ \boldsymbol{O}_3 & 2 \boldsymbol{E}^T \boldsymbol{J}_{\varphi} \end{pmatrix} \tag{19}$$

where $\boldsymbol{O}_3$ is the 3×3 zero matrix. Its time derivative is

$$\dot{\boldsymbol{J}}_o = \begin{pmatrix} \boldsymbol{O}_3 & \boldsymbol{O}_3 \\ \boldsymbol{O}_3 & 2 \left(\dot{\boldsymbol{E}}^T \boldsymbol{J}_{\varphi} + \boldsymbol{E}^T \dot{\boldsymbol{J}}_{\varphi} \right) \end{pmatrix}. \tag{20}$$

We also need the Jacobian of the l-th link expressed in $\mathcal{F}_c$:

$$\boldsymbol{J}_l = \begin{pmatrix} \boldsymbol{R}_o & \boldsymbol{S}(\boldsymbol{p}_o) \boldsymbol{R}_o \\ \boldsymbol{O}_3 & \boldsymbol{R}_o \end{pmatrix} \boldsymbol{J}_l^o = \boldsymbol{V}_o^c \boldsymbol{J}_l^o \tag{21}$$

where $\boldsymbol{J}_l^o$ is the Jacobian of the link in $\mathcal{F}_o$ and $\boldsymbol{V}_o^c$ camera-object velocity twist transformation. It is $\dot{\boldsymbol{J}}_l = \dot{\boldsymbol{V}}_o^c \boldsymbol{J}_l^o + \boldsymbol{V}_o^c \dot{\boldsymbol{J}}_l^o$. Thus, $\boldsymbol{J}_i$ and $\dot{\boldsymbol{J}}_i$ can be composed as follows:

$$\boldsymbol{J}_i = \begin{cases} (\boldsymbol{J}_o \quad \boldsymbol{O}_{6 \times n}) & \text{if } \boldsymbol{\ell}_i \in \text{FB}, \\ (\boldsymbol{J}_o \quad \boldsymbol{J}_l \quad \boldsymbol{O}_{6 \times (n-l)}) & \text{if } \boldsymbol{\ell}_i \in l\text{-th link}, \end{cases} \tag{22}$$

$$\dot{\boldsymbol{J}}_i = \begin{cases} (\dot{\boldsymbol{J}}_o \quad \boldsymbol{O}_{6 \times n}) & \text{if } \boldsymbol{\ell}_i \in \text{FB}, \\ (\dot{\boldsymbol{J}}_o \quad \dot{\boldsymbol{J}}_l \quad \boldsymbol{O}_{6 \times (n-l)}) & \text{if } \boldsymbol{\ell}_i \in l\text{-th link}. \end{cases} \tag{23}$$

Finally, the derivative of $\boldsymbol{L}_i$ (eq. (3)) and the variables needed for its computation, are in (24)–(26) at the top of the page.

REFERENCES

[1] J. Vaillant, K. Bouyarmane, and A. Kheddar, "Multi-character physical and behavioural interactions controller," *IEEE Trans. Visual. Comput. Graphics*, vol. 23, no. 6, pp. 1650–1662, 2017.

[2] K. Bouyarmane, J. Vaillant, K. Chappellet, and A. Kheddar, "Multi-robot and force task-space control with quadratic programming," *IEEE Trans. Robot.*, submitted.

[3] G. Park, A. Argyros, and W. Woo, "Efficient 3D hand tracking in articulation subspaces for the manipulation of virtual objects," in *33rd Computer Graphics International*, 2016, pp. 33–36.

[4] Z. Cao, T. Simon, S.-E. Wei, and Y. Sheikh, "Realtime multi-person 2D pose estimation using part affinity fields," in *IEEE Conf. on Computer Vision and Pattern Recognition*, 2017, pp. 1302–1310.

[5] T. Schmidt, K. Hertkorn, R. Newcombe, Z. Marton, M. Suppa, and D. Fox, "Depth-based tracking with physical constraints for robot manipulation," in *IEEE Int. Conf. on Robotics and Automation*, 2015, pp. 119–126.

[6] J. Sturm, C. Stachniss, and W. Burgard, "A probabilistic framework for learning kinematic models of articulated objects," *Journal of Artificial Intelligence Research*, vol. 41, pp. 477–526, 2011.

[7] C. G. Cifuentes, J. Issac, M. Wüthrich, S. Schaal, and J. Bohg, "Probabilistic articulated real-time tracking for robot manipulation," *IEEE Robot. Autom. Lett.*, vol. 2, no. 2, pp. 577–584, 2017.

[8] M. Krainin, P. Henry, X. Ren, and D. Fox, "Manipulator and object tracking for in-hand 3D object modeling," *Int. J. Robot. Res.*, vol. 30, no. 11, pp. 1311–1327, 2011.

[9] R. Martín Martín, S. Höfer, and O. Brock, "An integrated approach to visual perception of articulated objects," in *IEEE Int. Conf. on Robotics and Automation*, 2016, pp. 5091–5097.

[10] M. Klingensmith, T. Galluzzo, C. Dellin, M. Kazemi, J. A. D. Bagnell, and N. Pollard , "Closed-loop servoing using real-time markerless arm tracking," in *Humanoids Workshop at IEEE ICRA*, 2013.

[11] K. Nickels and S. Hutchinson, "Model-based tracking of complex articulated objects," *IEEE Trans. Robot. Automat.*, vol. 17, no. 1, pp. 28–36, 2001.

[12] T. Drummond and R. Cipolla, "Real-time visual tracking of complex structures," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 24, no. 7, pp. 932–946, 2002.

[13] A. I. Comport, E. Marchand, and F. Chaumette, "Kinematic sets for real-time robust articulated object tracking," *Image and Vision Computing*, vol. 25, no. 3, pp. 374–391, 2007.

[14] A. I. Comport, E. Marchand, M. Pressigout, and F. Chaumette, "Real-time markerless tracking for augmented reality: the virtual visual servoing framework," in *IEEE Trans. Visual. Comput. Graphics*, vol. 12, no. 4, 2006, pp. 615–628.

[15] X. Gratal, J. Romero, and D. Kragic, "Virtual visual servoing for real-time robot pose estimation," *IFAC Proceedings Volumes*, vol. 44, no. 1, pp. 9017–9022, 2011.

[16] A. Paolillo, A. Bolotnikova, K. Chappellet, and A. Kheddar, "Visual estimation of articulated objects configuration during manipulation with a humanoid," in *IEEE/SICE Int. Symp. on Syst. Integration*, 2017, pp. 330–335.

[17] F. Chaumette and S. Hutchinson, "Visual Servo Control, Part I: Basic Approaches," *IEEE Robot. Automat. Mag.*, vol. 13, no. 4, pp. 82–90, 2006.

[18] F. S. Grassia, "Practical parameterization of rotations using the exponential map," *J. of Graphics Tools*, vol. 3, no. 3, pp. 29–48, 1998.

[19] I. Sa, S. Hrabar, and P. Corke, "Inspection of pole-like structures using a visual-inertial aided VTOL platform with shared autonomy," *Sensors*, vol. 15, no. 9, pp. 22 003–22 048, 2015.

[20] X. Xinjilefu, S. Feng, and C. G. Atkeson, "Dynamic state estimation using quadratic programming," in *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, 2014, pp. 989–994.

[21] D. J. Agravante, G. Claudio, F. Spindler, and F. Chaumette, "Visual servoing in an optimization framework for the whole-body control of humanoid robots," *IEEE Robot. Autom. Lett.*, vol. 2, no. 2, pp. 608–615, 2017.

[22] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *IEEE Conf. on Computer Vision and Pattern Recognition*, vol. 1, 2005, pp. 886–893.

[23] V. Kazemi and J. Sullivan, "One millisecond face alignment with an ensemble of regression trees," in *IEEE Conf. on Computer Vision and Pattern Recognition*, 2014, pp. 1867–1874.

[24] 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, 2014.