An Image-based Trajectory Planning Approach for Robust Robot Programming by Demonstration

Aleksandar Vakanski, Farrokh Janabi-Sharifi, *Senior Member, IEEE*, and Iraj Mantegh

***Abstract*—The article proposes a new robot programming-by-demonstration framework, which integrates a visual servoing tracking control to robustly follow a trajectory generated from observed demonstrations. The constraints originating from the use of a visual servoing controller are incorporated into the trajectory learning phase, to guarantee feasibility of the generated plan for task execution. The observational learning is solved as a constrained optimization problem, with an objective to generalize from a set of trajectories of salient features in the image space of a vision camera. The proposed approach is evaluated experimentally for learning trajectories acquired from kinesthetic demonstrations.**

***Keywords*—Cognitive robotics, visual servoing, programming by demonstration.**

I. Introduction

ROBOT Programming by Demonstration (PbD) is a recent trend in robotics, employed to transfer new skills to robots from observations of tasks demonstrated by humans or other robots [1–3]. A typical robot PbD learning process consists of observing the demonstrations, followed by task modeling and planning steps, leading to task execution by the robot learner [4]. Despite the applicability of different types of sensors for *task perception* [2, 4–6], this work employs a vision sensor, due to the non-intrusive character of the vision-based measurements. An important aspect of the PbD process that has received little attention is the step of *task execution*, being often considered as an independent step, for which it is assumed that different types of robot control laws can be employed [7]. In practice however, the type of controller can impose certain constraints on the system, which might render the obtained reproduction strategy unsuitable for achieving the task goal. For example, the required command velocities and/or accelerations might not be realizable by the robot learner, or the required motions might violate the robot's dexterous workspace limits. Therefore, in this work both task learning and execution (control) are considered *synergistically*.

The PbD problem here is formulated as a constrained optimization problem, where the objective is to find an optimal reproduction strategy with respect to a given cost function, subject to constraints imposed by not only the task but also the selected controller. From the control perspective, the uncertainties in the execution step can cause incorrect positioning of the robot end-point with respect to the scene objects. For instance, joints' gear backlashes or slippages, bending of the robot links, poor fixturing of the objects, or incorrect estimation of the poses (positions and orientations) of scene objects from noisy sensor measurements can all have unfavorable effects on system's performance. Under these circumstances, the execution of the generated PbD strategy can fail to correctly reproduce the desired robot behavior. To address the robustness of robots positioning under uncertainties during the task execution, we propose to employ a vision-based control strategy (i.e., visual servoing) [8–10]. The use of visual feedback information in the PbD literature is mostly limited to look-and-move control, meaning that vision cameras are used for extracting the positions and/or orientations of objects in the scene, but they are not used directly for closed-loop control of robots' motion. The presented work formulates and analyzes synergistic integration of visual servo control into a PbD framework. The synergy originates from the design of perception, planning and execution steps to ensure improved overall system performance and enhanced robustness to modeling and measurement errors.

The proposed approach employs a set of demonstrations captured as trajectories of relevant scene features projected into the image plane of a stationary camera. The Kalman smoothing algorithm [11] is initially employed to extract a generalized trajectory for each image feature. This set represents smooth and continuous averages of the observed feature trajectories, to be used as reference trajectories in generating a plan for task reproduction. Similarly, a Kalman smoother is employed for recovering reference velocities of the tracked object from the demonstrations. The planning step is formulated as an optimization problem, with a cost function

Manuscript submitted September 19, 2016. This work was supported through a collaborative research and development grant from the Natural Sciences and Engineering Research Council of Canada CRDPJ 350266 – 07. The work was also supported by the Ryerson Equipment Fund.

A. Vakanski (corresponding author) is with the Industrial Technology program at the University of Idaho (address: 1776 Science Center Drive, Idaho Falls, ID, 83402, USA; e-mail: vakanski@idaho.edu, phone: 1-208-757-5422).

F. Janabi-Sharifi is with the Department of Mechanical and Industrial Engineering at Ryerson University, Toronto, Canada (fsharifi@ryerson.ca).

I. Mantegh is with the National Research Council Canada - Institute for Aerospace Research (NRC-IAR), Montreal, Canada (iraj.mantegh@cnrc-nrc.gc.ca).

which minimizes the distance between the current and reference image feature vectors and current and reference object velocities. The constraints in the model include the visual, workspace task and robot constraints (e.g., those related to the visibility of the features, workspace, robot kinematics, etc). All the constraints are formulated in a linear or conic form, thus enabling to solve the model as a convex optimization problem. Subsequently, an image-based visual servoing (IBVS) controller is employed to ensure robust execution of the generated feature trajectories in presence of uncertainties, such as image noise and camera modeling errors [8–10].

In the context of using a visual servo tracker, we implement planning of trajectories for a set of target features directly in the image space of a vision camera. Differently from the previous works on path planning in visual servoing [12–14], the planning step in the proposed framework is initialized by the available examples of the image feature trajectories that are acquired from the demonstrations. Since direct planning in the image space can cause sub-optimal trajectories of the robot's end-point in the Cartesian space, a constraint is formulated in the model which forces the respective Cartesian trajectory to stay within the envelope of the demonstrated motions.

There are several works in robot PbD that employ direct learning in the image space of a vision camera [7, 15–17]. The presented article differs from this body of works in the integration of the different learning steps and the different type of constraints for a robotic arm into a single framework, as opposed to path learning for a mobile robot [7, 17], or tool-centered learning from demonstrations [16].

The novelties of the proposed approach include: (i) introduction of a unique framework for PbD learning with all the steps of the PbD process, i.e., observation, planning and execution steps, taking place in the image space (of a robot's vision system); (ii) formulation of the planning process as a convex optimization problem to incorporate important task constraints, which are often neglected in the PbD domain; and (iii) synergistic integration of a vision-based control strategy into a PbD framework for robust task execution.

This article is organized as follows. Section II introduces the notation. Sections III and IV present the objective function and the constraints of the optimization model, respectively. The mathematical formulation of a second-order conic optimization model is provided in Section V. In Section VI the image-based visual tracker for following the generated feature trajectories is described. Section VII presents experimental evaluation of the approach. Section VIII provides discussion of the obtained results. Finally, Section IX concludes the work.

II. Preliminaries

It is assumed that a task is demonstrated by a human teacher M times in front of a robot learner. The robot observes the demonstrations via a stationary vision camera. Through processing the sequences of recorded images from the demonstrations, the task is described by image-space trajectories of several salient features from the environment. The object of interest in the scene depends on the task, and it can be a workpiece, a tool, demonstrator's hand, etc. Among the different type of features that can be extracted from the images, this work utilizes coordinates of points in the image plane of the camera (e.g., corners, area centroids, etc).

The observed pixel coordinates of the feature point n at time instant t_k for the demonstration m are denoted by $\mathbf{u}_n^{(m)}(t_k) = \left[u_n^{(m)}(t_k) \quad v_n^{(m)}(t_k)\right]^T \in \mathbb{R}^2$, for $n = 1,2, \ldots,N$ and $m = 1,2, \ldots,M$, where N denotes the total number of used feature points, and M pertains to the total number of recorded demonstrations. We assume that a calibrated pinhole camera is used, i.e., the pixel coordinates of the features points can be transformed into spatial image plane coordinates [9]

$$\begin{cases} x_n^{(m)}(t_k) = \left(u_n^{(m)}(t_k) - u_0\right) / f k_u \\ y_n^{(m)}(t_k) = \left(v_n^{(m)}(t_k) - v_0\right) / f k_v \end{cases}, \tag{1}$$

where u_0 and v_0 are the coordinates of the principal point, f denotes the focal length of the camera, and k_u and k_v are the horizontal and vertical number of pixels per unit length of the vision sensor. The pairs of image plane coordinates for the feature point n at time t_k are denoted by $\mathbf{p}_n^{(m)}(t_k) = \left[x_n^{(m)}(t_k) \quad y_n^{(m)}(t_k)\right]^T \in \mathbb{R}^2$. The set of all observed features for the demonstration m forms the image features parameters vector, with the following notation used $\mathbf{s}^{(m)}(t_k) = \left[\mathbf{p}_1^{(m)}(t_k)^T \quad \mathbf{p}_2^{(m)}(t_k)^T \quad \cdots \quad \mathbf{p}_n^{(m)}(t_k)^T\right] \in \mathbb{R}^{2n}$.

A graphical representation of the environment is shown in Fig. 1. It depicts the robot, the camera that is fixed in the workspace, and the object, which is manipulated either by a human operator during the demonstrations or by a robot during the task reproduction. The notation for the coordinate frames used is introduced in the figure. The positions and orientations of the coordinate frame i with respect to frame j are denoted by $\left(\mathbf{P}_i^j, \mathbf{R}_i^j\right) \in \mathrm{SE}(3)$. Note that the set of Euler roll-pitch-yaw angles $\boldsymbol{\phi}_i^j$ will also be used for representation of the orientation whenever required.

III. Optimization Model – Objective Function

Based on the set of M observed image feature trajectories from the demonstrations (i.e., $\mathbf{s}^{(m)}(t_k)$ for $m = 1,2,\ldots,M$ and $k = 1,2,\ldots$), the goal is to retrieve a generalized trajectory of the image features $\mathbf{s}(t_k)$ for $k = 1,2,...,g$, which will allow the robot learner to reproduce the demonstrated task. The notation g is used for the number of time frames related to the duration of the generalized trajectory for task reproduction.

There are several challenges in generating a task reproduction trajectory directly in the image space [12]. Namely, small displacements of the feature parameters in the image can result in high velocities of the feature points in the Cartesian space. Thus, mapping of the image features trajectories into the Cartesian space can lead to sub-optimal

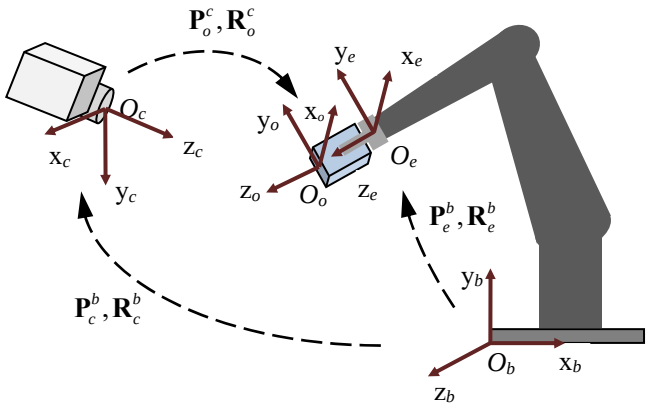


Fig. 1. The robot learning cell, consisting of a camera with a frame $\mathcal{F}_c(O_c, x_c, y_c, z_c)$, a manipulated object $\mathcal{F}_o(O_o, x_o, y_o, z_o)$, robot end-point frame $\mathcal{F}_e(O_e, x_e, y_e, z_e)$, and a robot base frame $\mathcal{F}_b(O_b, x_b, y_b, z_b)$. The respective transformations $(\mathbf{P}_i^j, \mathbf{R}_i^j)$ between the coordinate frames i and j are shown in the figure.

Cartesian trajectories, which might violate the workspace limits or cause collisions with the objects in the environment. To avoid such scenarios, the generation of a task reproduction trajectory is solved here as a constrained optimization problem. The objective function is formulated for simultaneous optimization of the image features trajectories and the velocity of the object of interest with regards to distance functions.

For that purpose, first reference image trajectories are generated. Based on the set of demonstrated trajectories for each feature point n (i.e., $\{\mathbf{p}_n^{(1)}, \mathbf{p}_n^{(2)}, \ldots, \mathbf{p}_n^{(M)}\}$), a Kalman smoother [11] is used to obtain a smooth and continuous reference trajectory $\mathbf{p}_n^{\text{ref}}$. A brief explanation of the Kalman smoother algorithm is provided in Appendix A. The observed state of each Kalman smoother in (A1) is formed by concatenation of the measurements from all demonstrations, that is $\mathbf{o}_k = [\mathbf{p}_n^{(1)}(t_k)^T \quad \mathbf{p}_n^{(2)}(t_k)^T \quad \ldots \quad \mathbf{p}_n^{(M)}(t_k)^T]^T \in R^{2M}$. The combined reference trajectories of the feature points form the reference feature parameter vector $\mathbf{s}^{\text{ref}}(t_k)$, for $k = 1, 2, \ldots, g$. Subsequently, the first part of the objective function at the time instant t_k is formulated to minimize the sum of distances between the image features parameters and the reference image features parameters at the next time instant, i.e., $\left\|\mathbf{p}_n(t_{k+1}) - \mathbf{p}_n^{\text{ref}}(t_{k+1})\right\|$. The goal is to generate continuous feature trajectories in the image space, i.e., to prevent sudden changes in the trajectories. The notation $\|\cdot\|$ is used to denote Euclidean norm of a vector. To define the optimization over a conic set of variables, a set of auxiliary variables is introduced as $\tau_n \geq \left\|\mathbf{p}_n(t_{k+1}) - \mathbf{p}_n^{\text{ref}}(t_{k+1})\right\|$, for each feature point $n = 1, 2, \ldots, N$.

The second part of the objective function pertains to the velocity of the target object. The goal is to ensure that the image trajectories are mapped to smooth and continuous velocities of the manipulated object. To retrieve the velocity of the object from camera acquired images, first the pose of the object at each time instant is extracted. Here it is assumed that a geometric model of the target object is available providing knowledge about the 3D distances between the object's feature points. The homography transformation between the features locations in an image and its corresponding 3D coordinates is utilized for pose extraction of the object with respect to the camera [18]. For estimation of the homography matrix, the correspondences of at least 4 coplanar points or 8 non-coplanar points in the acquired images are required. The pose of the object relative to the camera frame for the demonstration m is denoted by $\{\mathbf{P}_o^{c\,(m)}, \boldsymbol{\phi}_o^{c\,(m)}\}$, where $\mathbf{P}_o^c$ refers to the translational coordinates, and $\boldsymbol{\phi}_o^c$ denotes the Euler roll-pitch-yaw angles representation of the object's orientation in the camera frame. By differentiating the pose, the linear and angular velocity of the object in the camera frame $(\mathbf{v}_o^{c(m)}(t_k) \in \mathbb{R}^3, \boldsymbol{\omega}_o^{c(m)}(t_k) \in \mathbb{R}^3)$ at each time instant are obtained [19]. Similarly to the first part of the objective function, Kalman smoothers are employed to generate smooth averages of the linear and angular velocities of the object, i.e., $\mathbf{v}_o^{c,\,\text{ref}} = (\mathbf{v}_o^{c,\,\text{ref}}, \boldsymbol{\omega}_o^{c,\,\text{ref}})$. The optimization objective is formulated to minimize the sum of Euclidean distances between an unknown vector related to the current linear and angular velocities and the reference linear and angular velocities. By analogy to the first part of the objective function, two auxiliary conic variables are introduced: $\tau_{\mathbf{v}} \leq \left\|\mathbf{v}_o^c(t_k) - \mathbf{v}_o^{c,\,\text{ref}}(t_k)\right\|$ and $\tau_{\boldsymbol{\omega}} \leq \left\|\boldsymbol{\omega}_o^c(t_k) - \boldsymbol{\omega}_o^{c,\,\text{ref}}(t_k)\right\|$, respectively.

The objective function is then defined as a weighted minimization of the sum of variables $\tau_1, \ldots, \tau_N, \tau_{\mathbf{v}}, \tau_{\boldsymbol{\omega}}$, i.e.,

$$\text{minimize}\left\{\sum_{n=1}^{N} \alpha_n \tau_n + \alpha_{\mathbf{v}} \tau_{\mathbf{v}} + \alpha_{\boldsymbol{\omega}} \tau_{\boldsymbol{\omega}}\right\} \qquad (2)$$

where the α's coefficients are the weights of relative importance of the individual components in the cost function. The selection of the weighting coefficients α's in (2), and other implementation details, will be discussed in Section VII that is dedicated to experimental validation of the method.

To recapitulate, when one performs trajectory planning directly in the image space for each individual feature point, the generated set of image trajectories of the object's feature points might not translate into a feasible Cartesian trajectory of the object. In our case, the reference image feature vectors obtained with the Kalman smoothing do not necessarily map into a meaningful Cartesian pose of the object. Therefore, the optimization procedure is performed to ensure that the model variables are constrained such that at each time instant there exists a meaningful mapping between the feature parameters in the image space and the object's pose in the Cartesian space.

Thus starting from a set of reference feature parameters $\mathbf{s}^{\text{ref}}(t_{k+1}) = \{\mathbf{p}_n^{\text{ref}}(t_{k+1})\}_{n=1}^N$ and reference velocity $\mathbf{v}_o^{c,\,\text{ref}}(t_k)$, the optimization at each time instant t_k results in a set of image feature parameters $\mathbf{s}(t_{k+1})$ that is close to the reference image feature parameters $\mathbf{s}^{\text{ref}}(t_{k+1})$, and that entails feasible and

smooth Cartesian object velocity $\mathbf{v}_o^c(t_k)$. From the robot control perspective, the goal is to find an optimal velocity of the end-point (and subsequently, the velocity of object that is grasped by robot's gripper) $\mathbf{v}_o^c(t_k)$, which when applied at the current time will result in a feasible location of the image features at the next time step $\mathbf{s}(t_{k+1})$ (where the optimal location depends on the selection of the weighting coefficients in (2)).

IV. Optimization Model - Constraints

The following constraints are considered.

A. *Image space constraints*

I. The relationship between image features velocities and the velocity of the object in the camera frame is [8–10]

$$\mathbf{s}(t) = \mathbf{L}(t)\,\mathbf{v}_o^c(t)\ . \tag{3}$$

Using the Euler forward discretization, (3) can be written as

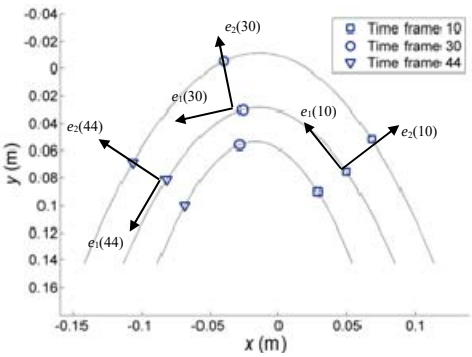
$$\mathbf{s}(t_{k+1}) = \mathbf{s}(t_k) + \mathbf{L}(t_k)\,\mathbf{v}_o^c(t_k)\Delta t_k\ , \tag{4}$$

where Δt_k denotes the sampling period at time t_k, and $\mathbf{L}(t_k)$ in the literature is often called image Jacobian matrix, or interaction matrix [8]. The derivation of the interaction matrix for the considered camera-object configuration is given in Appendix B.

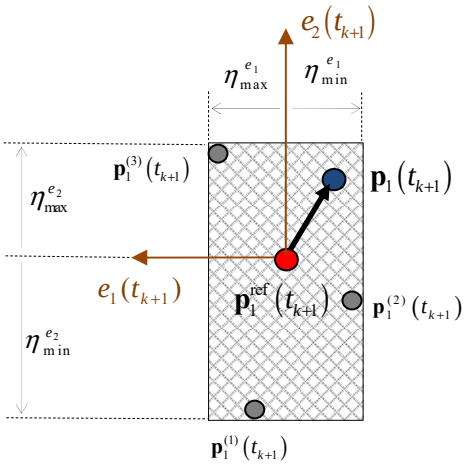
II. The second constraint ensures that the features parameters in the next time instant $\mathbf{s}(t_{k+1})$ are within the bounds of the demonstrated trajectories. For that purpose, first at each time step we find the principal directions of the demonstrated features, by extracting the eigenvectors of the covariance matrix of the demonstrations. For instance, for the feature 1, the covariance matrix at each time instant is associated with concatenated observation vectors from the set of demonstrations, i.e., $\mathrm{cov}\left(\mathbf{p}_1^{(1)}(t_k), \mathbf{p}_1^{(2)}(t_k), \ \ , \mathbf{p}_1^{(M)}(t_k)\right)$. For a set of three trajectories in Fig. 2a, the eigenvectors $\mathbf{e}_1$, $\mathbf{e}_2$ are illustrated at different time instants t_k (k = 10, 30, 44).

At time instant t_k, the matrix of eigenvectors $\mathbf{E}_p(t_k)$ rotates the observation vectors along the principal directions of the demonstrated motions. For instance, the observed parameters for the feature 1 in the three demonstrations in the next time instant $\left(\mathbf{p}_1^{(1)}(t_{k+1}), \mathbf{p}_1^{(2)}(t_{k+1}), \text{ and } \mathbf{p}_1^{(3)}(t_{k+1})\right)$, are shown rotated in Fig. 2b with respect to the reference image feature parameters $\mathbf{p}_1^{\mathrm{ref}}(t_{k+1})$. The rotated vectors $\mathbf{p}_1^{(m)}(t_{k+1}) - \mathbf{p}_1^{\mathrm{ref}}(t_{k+1})$ for $m = 1, 2, 3$ define the boundaries of the demonstrated space at time instant t_{k+1}, which corresponds to the hatched section in Fig. 2b. The inner and outer bounds of the demonstrated envelope are found as:

$$\begin{cases} \boldsymbol{\eta}_{\max}(t_{k+1}) = \max\limits_{m=1,2,\ldots,M} \left(\mathbf{E}_p(t_{k+1})\left(\mathbf{p}_1^{(m)}(t_{k+1}) - \mathbf{p}_1^{\mathrm{ref}}(t_{k+1})\right)\right) \\ \boldsymbol{\eta}_{\min}(t_{k+1}) = \min\limits_{m=1,2,\ldots,M} \left(\mathbf{E}_p(t_{k+1})\left(\mathbf{p}_1^{(m)}(t_{k+1}) - \mathbf{p}_1^{\mathrm{ref}}(t_{k+1})\right)\right) \end{cases}. \tag{5}$$



(a)



(b)

Fig. 2. (a) The eigenvectors of the covariance matrix $\mathbf{e}_1(t_k)$, $\mathbf{e}_2(t_k)$ for three demonstrations at times k = 10, 30 and 44. The observed image plane features are depicted by different type of marks; (b) The observed parameters for the feature 1, $\mathbf{p}_1^{(1)}(t_{k+1})$, $\mathbf{p}_1^{(2)}(t_{k+1})$, $\mathbf{p}_1^{(3)}(t_{k+1})$, are rotated by the eigenvector matrix $\mathbf{E}_p(t_{k+1})$. The vector $\mathbf{p}_1(t_{k+1}) - \mathbf{p}_1^{\mathrm{ref}}(t_{k+1})$ is required to belong to the region bounded by $\eta_{\min}(t_{k+1})$ and $\eta_{\max}(t_{k+1})$.

The maximum and minimum operations in (5) are performed separately for the horizontal and vertical image coordinates, so that the bounds $\boldsymbol{\eta}_{\max/\min} = \left[\eta_{\max/\min}^{e_1} \quad \eta_{\max/\min}^{e_2}\right]^T$ represent 2×1 vectors (see Fig. 2b).

Suppose there exists an unknown distance vector $\mathbf{p}_1(t_{k+1}) - \mathbf{p}_1^{\mathrm{ref}}(t_{k+1})$, and its coordinate transformation when rotated in the instantaneous demonstrated direction, is denoted

$$\boldsymbol{\eta}(t_{k+1}) = \mathbf{E}_p(t_{k+1})\left(\mathbf{p}_1(t_{k+1}) - \mathbf{p}_1^{\mathrm{ref}}(t_{k+1})\right). \tag{6}$$

Then the following constraint ensures that this variable is bounded within the demonstrated envelope, i.e.,

$$\boldsymbol{\eta}_{\min}(t_{k+1}) \le \boldsymbol{\eta}(t_{k+1}) \le \boldsymbol{\eta}_{\max}(t_{k+1})\ . \tag{7}$$

Note that the inequalities in (7) operate element-wise for each dimension of the vectors. The same notation holds in the text that follows when vector inequalities are used.

By introducing non-negative excess and slack variables $\boldsymbol{\eta}_e$ and $\boldsymbol{\eta}_s$, respectively, the constraint (7) can be represented with two linear equalities:

$$\begin{cases} \boldsymbol{\eta}(t_{k+1}) + \boldsymbol{\eta}_e(t_{k+1}) = \boldsymbol{\eta}_{\max}(t_{k+1}) \\ \boldsymbol{\eta}(t_{k+1}) - \boldsymbol{\eta}_s(t_{k+1}) = \boldsymbol{\eta}_{\min}(t_{k+1}) \end{cases}. \tag{8}$$

III. The image features trajectories should also stay in the field-of-view of the camera. Therefore, if the image boundaries are denoted as horizontal image limits $p^{x,\max}$ and $p^{x,\min}$ and vertical image limits $p^{y,\max}$ and $p^{y,\min}$, then at each time instant the following set should hold:

$$\begin{cases} p^{x,\min} \le p_n^x(t_k) \le p^{x,\max} \\ p^{y,\min} \le p_n^y(t_k) \le p^{y,\max} \end{cases} \quad \text{for} \quad n = 1,2,...,N, \tag{9}$$

or with adding non-negative excess and slack variables, the constraints in (9) are rewritten as

$$\begin{cases} \mathbf{p}_n(t_k) + \mathbf{p}_e(t_k) = \mathbf{p}_{\max} \\ \mathbf{p}_n(t_k) - \mathbf{p}_s(t_k) = \mathbf{p}_{\min} \end{cases} \quad \text{for} \quad n = 1,2,...,N \tag{10}$$

This constraint may be redundant for most tasks, since the general assumption is that the demonstrated image trajectories are within the field-of-view of the camera. However, the constraint might be useful if the trajectories are close to the boundaries of the image. For instance, the image limits $\mathbf{p}_{\min}$ and $\mathbf{p}_{\max}$ can be set to 5 or 10 pixels from the field-of-view, which can prevent the executable trajectories to get very close to the image boundaries, and it will reduce the chances of losing some features during the visual tracking, due to image noise or other uncertainties.

B. Cartesian space constraints

I. The first constraint relates the Cartesian trajectory position with the velocity of the object with respect to the camera frame:

$$\frac{d}{dt}\mathbf{P}_o^c(t) = \mathbf{v}_o^c(t) \tag{11}$$

or in a discrete form

$$\mathbf{P}_o^c(t_{k+1}) = \mathbf{P}_o^c(t_k) + \mathbf{v}_o^c(t_k)\Delta t_k \ . \tag{12}$$

II. The next important constraint is to ensure that the Cartesian trajectory of the object stays within the demonstrated space. This constraint will prevent potential collisions of the object with the surrounding environment, under assumption that the demonstrated space is free of obstacles.

Similarly to the image based constraint II in (5–8), the inner and outer bounds of the demonstrations are found from the principal directions of the covariance matrix of the demonstrated Cartesian trajectories, i.e.,

$$\begin{cases} \boldsymbol{\mu}_{\max}(t_{k+1}) = \max\limits_{m=1,2,...,M} \left(\mathbf{E}_P(t_{k+1}) \left(\mathbf{P}_o^{c(m)}(t_{k+1}) - \mathbf{P}_o^{c,\mathrm{ref}}(t_{k+1}) \right) \right) \\ \boldsymbol{\mu}_{\min}(t_{k+1}) = \min\limits_{m=1,2,...,M} \left(\mathbf{E}_P(t_{k+1}) \left(\mathbf{P}_o^{c(m)}(t_{k+1}) - \mathbf{P}_o^{c,\mathrm{ref}}(t_{k+1}) \right) \right) \end{cases}. \tag{13}$$

The value of the rotated distance vector in the next time instant is

$$\boldsymbol{\mu}(t_{k+1}) = \mathbf{E}_P(t_{k+1}) \left(\mathbf{P}_o^c(t_{k+1}) - \mathbf{P}_o^{c,\mathrm{ref}}(t_{k+1}) \right) \tag{14}$$

and it should be bounded by

$$\boldsymbol{\mu}_{\min}(t_{k+1}) \le \boldsymbol{\mu}(t_{k+1}) \le \boldsymbol{\mu}_{\max}(t_{k+1}) \, . \tag{15}$$

By introducing non-negative excess $\boldsymbol{\mu}_e$ and slack $\boldsymbol{\mu}_s$ variables, the constraint can be represented as equalities:

$$\begin{cases} \boldsymbol{\mu}(t_{k+1}) + \boldsymbol{\mu}_e(t_{k+1}) = \boldsymbol{\mu}_{\max}(t_{k+1}) \\ \boldsymbol{\mu}(t_{k+1}) - \boldsymbol{\mu}_s(t_{k+1}) = \boldsymbol{\mu}_{\min}(t_{k+1}) \end{cases}. \tag{16}$$

III. Another constraint is introduced for the velocity of the object, which is bounded between $\mathbf{v}_{\min}$ and $\mathbf{v}_{\max}$ at each time step. These values could correspond to the extreme values of the velocities that can be exerted by the robot's end-point,

$$\mathbf{v}_{\min} \le \mathbf{v}_o(t_k) \le \mathbf{v}_{\max} \, , \tag{17}$$

or with introducing non-negative excess and slack variables we have

$$\begin{cases} \mathbf{v}_o(t_k) + \mathbf{v}_e(t_k) = \mathbf{v}_{\max} \\ \mathbf{v}_o(t_k) - \mathbf{v}_s(t_k) = \mathbf{v}_{\min} \end{cases}. \tag{18}$$

C. Robot manipulator constraints

I. The first constraint relates the robot joint variables to the object's velocity. It is assumed that the object is grasped in the robot's gripper (Fig. 1), with the velocity transformation between the object frame $\mathbf{v}_o^b = \left(\mathbf{v}_o^b, \boldsymbol{\omega}_o^b \right)$ and robot's end-point frame $\mathbf{v}_e^b$ given by

$$\begin{cases} \mathbf{v}_o^b = \mathbf{v}_e^b + \boldsymbol{\omega}_e^b \times \mathbf{P}_{e,o}^b = \mathbf{v}_e^b - S\left(\mathbf{R}_e^b \, \mathbf{P}_o^e \right) \boldsymbol{\omega}_e^b \\ \boldsymbol{\omega}_o^b = \boldsymbol{\omega}_e^b \end{cases}. \tag{19}$$

In the above equation, the notation $\mathbf{P}_{e,o}^b$ is used to describe the coordinates of the position vector of the object frame with respect to the robot's end-point frame, expressed in the base frame.

The notation $S(\cdot)$ in (19) denotes a skew-symmetric matrix, which for an arbitrary vector $\boldsymbol{a} = [a_x, a_y, a_z]$ is defined as

$$S(\boldsymbol{a}) = \begin{bmatrix} 0 & -a_z & a_y \\ a_z & 0 & -a_x \\ -a_y & a_x & 0 \end{bmatrix}. \tag{20}$$

The differential kinematic equation of the robot is given by

$$\mathbf{v}_e^b(t) = \mathbf{J}\left(\mathbf{q}(t)\right)\dot{\mathbf{q}}(t)\ , \tag{21}$$

where $\mathbf{q}$ is a $\xi\times 1$ vector of robot joint variables, and $\mathbf{J}\left(\mathbf{q}(t)\right)$ is the robot Jacobian matrix in the robot base frame. Hence, the relationship between the joint variables and the object velocity in the camera frame is obtained using (19) and (21)

$$\begin{aligned}\dot{\mathbf{q}}(t) &= \mathbf{J}^{\dagger}\left(\mathbf{q}(t)\right)\mathbf{v}_e^b(t)\\ &= \mathbf{J}^{\dagger}\left(\mathbf{q}(t)\right)\begin{bmatrix}\mathbf{I}_{3\times3} & -S\left(\mathbf{R}_e^b(t)\,\mathbf{P}_o^e\right)\\ \mathbf{0}_{3\times3} & \mathbf{I}_{3\times3}\end{bmatrix}^{-1}\mathbf{v}_o^b(t)\\ &= \mathbf{J}^{\dagger}\left(\mathbf{q}(t)\right)\begin{bmatrix}\mathbf{I}_{3\times3} & -S\left(\mathbf{R}_e^b(t)\,\mathbf{P}_o^e\right)\\ \mathbf{0}_{3\times3} & \mathbf{I}_{3\times3}\end{bmatrix}^{-1}\begin{bmatrix}\mathbf{R}_c^b & \mathbf{0}_{3\times3}\\ \mathbf{0}_{3\times3} & \mathbf{R}_c^b\end{bmatrix}\mathbf{v}_o^c(t),\end{aligned} \tag{22}$$

where $\mathbf{I}_{3\times3}$ and $\mathbf{0}_{3\times3}$ are 3×3 identity and zeroes matrices respectively, and $\mathbf{J}^{\dagger}\left(\mathbf{q}(t)\right)\in\mathbb{R}^{\xi\times6}$ denotes the pseudo-inverse of the robot Jacobian matrix. At time t_k, the equation (22) can be represented in a discrete form

$$\mathbf{q}(t_{k+1}) = \mathbf{q}(t_k) + \mathbf{J}^{\dagger}\left(\mathbf{q}(t_k)\right)\begin{bmatrix}\mathbf{I} & -S\left(\mathbf{R}_e^b(t_k)\,\mathbf{P}_o^e\right)\\ \mathbf{0}_{3\times3} & \mathbf{I}\end{bmatrix}^{-1}\begin{bmatrix}\mathbf{R}_c^b & \mathbf{0}_{3\times3}\\ \mathbf{0}_{3\times3} & \mathbf{R}_c^b\end{bmatrix}\mathbf{v}_o^c(t_k)\,\Delta t_k\,. \tag{23}$$

The rotation matrix of robot's end-point in base frame $\mathbf{R}_e^b(t_k)$ is obtained by using the robot's forward kinematics. The rotation of the camera frame in robot base frame $\mathbf{R}_c^b$ is found from the camera calibration.

II. A constraint for the robot joint variables (to be within the limits) is defined as

$$q_{\min}^{\varsigma} \le q^{\varsigma}(t_{k+1}) \le q_{\max}^{\varsigma} \quad \text{for } \varsigma = 1,2,...,\xi\,, \tag{24}$$

or in the form of equalities, (24) becomes

$$\begin{cases}\mathbf{q}(t_{k+1}) + \mathbf{q}_e(t_{k+1}) = \mathbf{q}_{\max}\\ \mathbf{q}(t_{k+1}) - \mathbf{q}_s(t_{k+1}) = \mathbf{q}_{\min}\end{cases} \text{for } \varsigma = 1,2,...,\xi\,. \tag{25}$$

V. Second Order Conic Optimization

The formulated problem with the given cost function and constraints is solved as a second-order conic optimization problem [20]. The optimization is defined as

$$\begin{aligned}&\text{minimize} && \mathbf{c}^T\mathbf{z}\\ &\text{subject to} && \mathbf{A}\mathbf{z} = \mathbf{b}\\ &&& \mathbf{z}_c \in \mathcal{K}\end{aligned} \tag{26}$$

where the inputs are a matrix $\mathbf{A}\in\mathbb{R}^{l\times d}$, vectors $\mathbf{c}\in\mathbb{R}^d$ and $\mathbf{b}\in\mathbb{R}^l$, and the output is the vector $\mathbf{z}\in\mathbb{R}^d$. The part of the vector $\mathbf{z}$ that corresponds to the conic constraints is denoted $\mathbf{z}_c$, whereas the part that corresponds to the linear constraints is denoted $\mathbf{z}_l$, that is $\mathbf{z} = \begin{bmatrix}\mathbf{z}_c^T & \mathbf{z}_l^T\end{bmatrix}^T$. For a vector variable $\mathbf{z}_{c,i} = \begin{bmatrix}z_{c,i}^1 & z_{c,i}^2 & \cdots & z_{c,i}^g\end{bmatrix}$ that belongs to a second-order cone $\mathcal{K}$, one has: $z_{c,i}^1 \le \left\|\begin{bmatrix}z_{c,i}^2 & z_{c,i}^3 & \cdots & z_{c,i}^g\end{bmatrix}\right\|$. An important engaging property of the conic optimization problems is the convexity of the solutions space, i.e., global convergence is warranted within the set of feasible solutions. To cast a problem into a second-order optimization requires a mathematical model expressed through linear or conic constraints.

In the considered case the linear equations (4), (6), (8), (10), (12), (14), (16), (18), (23) and (25) are combined to form the equality constraints in (26). The cost function defined in (2) operates over the conic variables:

$$\begin{cases}\tau_n \le \left\|\mathbf{p}_n(t_{k+1}) - \mathbf{p}_n^{\text{ref}}(t_{k+1})\right\|\ , \text{ for } n = 1,2,...,N\\ \tau_{\upsilon} \le \left\|\boldsymbol{\upsilon}_o^c(t_k) - \boldsymbol{\upsilon}_o^{c,\text{ref}}(t_k)\right\|\\ \tau_{\omega} \le \left\|\boldsymbol{\omega}_o^c(t_k) - \boldsymbol{\omega}_o^{c,\text{ref}}(t_k)\right\|\end{cases} \tag{27}$$

with the objective to minimize simultaneously the norms of the distances between the obtained and referenced image features trajectories and velocities.

Therefore, the optimization variable $\mathbf{z}$ in (26) at the time instant t_k is formed by concatenation of the variables from the conic constraints given in (27)

$$\begin{aligned}\mathbf{z}_c(t_k) = \Big[&\left[\tau_{\upsilon}, \boldsymbol{\upsilon}_o^c(t_k) - \boldsymbol{\upsilon}_o^{c,\text{ref}}(t_k)\right], \left[\tau_{\omega}, \boldsymbol{\omega}_o^c(t_k) - \boldsymbol{\omega}_o^{c,\text{ref}}(t_k)\right],\\ &\left[\tau_1, \mathbf{p}_1(t_{k+1}) - \mathbf{p}_1^{\text{ref}}(t_{k+1})\right], ..., \left[\tau_N, \mathbf{p}_N(t_{k+1}) - \mathbf{p}_N^{\text{ref}}(t_{k+1})\right]\Big]\end{aligned} \tag{28}$$

and the variables from the linear constraints

$$\begin{aligned}\mathbf{z}_l(t_k) = [&\mathbf{P}_o^c(t_{k+1}), \boldsymbol{\eta}(t_{k+1}), \boldsymbol{\mu}(t_{k+1}), \mathbf{q}(t_{k+1}),\\ &\mathbf{v}_e(t_k), \mathbf{v}_s(t_k), \mathbf{s}_e(t_{k+1}), \mathbf{s}_s(t_{k+1}), \boldsymbol{\eta}_e(t_{k+1}),\\ &\boldsymbol{\eta}_s(t_{k+1}), \boldsymbol{\mu}_e(t_{k+1}), \boldsymbol{\mu}_s(t_{k+1}), \mathbf{q}_e(t_{k+1}), \mathbf{q}_s(t_{k+1})]\end{aligned}\ , \tag{29}$$

i.e., $\mathbf{z}(t_k) = \begin{bmatrix}\mathbf{z}_c(t_k)^T & \mathbf{z}_l(t_k)^T\end{bmatrix}^T$. The total dimension of the vector $\mathbf{z}$ is $3\cdot(9+4N+\xi)+5+N$. From the cost function in (2), the part of the vector $\mathbf{c}$ in (26) that corresponds to the $\mathbf{z}_c(t_k)$ is

$$\mathbf{c}_c(t_k) = \begin{bmatrix}\alpha_{\upsilon} & \mathbf{0}_{1\times3} & \alpha_{\omega} & \mathbf{0}_{1\times3} & \alpha_1 & \mathbf{0}_{1\times2} & \cdots & \alpha_N & \mathbf{0}_{1\times2}\end{bmatrix}^T, \tag{30}$$

whereas the part $\mathbf{c}_l(t_k)$ corresponding to $\mathbf{z}_l(t_k)$ is all zeros, since those variables are not used in the cost function (2).

The known parameters for the optimization model at time t_k are: Δt_k, $\mathbf{s}^{\text{ref}}(t_{k+1})$, $\mathbf{v}_o^{c,\text{ref}}(t_k)$, $\mathbf{s}(t_k)$, $\mathbf{P}_o^c(t_k)$, $\mathbf{L}(t_k)$, $\mathbf{E}_p(t_{k+1})$, $\boldsymbol{\eta}_{\min}(t_{k+1})$, $\boldsymbol{\eta}_{\max}(t_{k+1})$, $\mathbf{E}_P(t_{k+1})$, $\boldsymbol{\mu}_{\min}(t_{k+1})$, $\boldsymbol{\mu}_{\max}(t_{k+1})$, $\mathbf{q}(t_k)$, $\mathbf{R}_e^b(t_k)$, $\mathbf{J}^{\dagger}(t_k)$, and the time independent parameters: $\mathbf{p}_{\min}$, $\mathbf{p}_{\max}$, $\mathbf{v}_{\min}$, $\mathbf{v}_{\max}$, $\mathbf{q}_{\min}$, $\mathbf{q}_{\min}$, $\mathbf{R}_c^b$, $\mathbf{P}_o^e$.

The optimization is solved in MATLAB by using the SeDuMi package [21].

VI. Image-based tracker

To follow the image feature trajectories $\mathbf{s}(t_{k+1})$ for $k=1,2,...,g$ generated from the optimization model, an image-based visual tracker is employed. This control ensures that the errors between the measured feature parameters $\overline{\mathbf{s}}(t)$ and the followed feature parameters $\mathbf{s}(t)$, i.e., $\mathbf{e}(t)=\overline{\mathbf{s}}(t)-\mathbf{s}(t)$, are driven to zero for $t\in(0,\infty)$. Selecting a controller for exponential decoupled decrease of the error $\mathbf{e}=-\lambda\mathbf{e}$, and using (3), one obtains

$$\mathbf{e}(t)=\overline{\mathbf{s}}-\mathbf{s}=\mathbf{L}(t)\,\mathbf{v}_o^c(t)-\mathbf{s}. \tag{31}$$

Hence

$$\mathbf{v}_o^c(t)=-\lambda\hat{\mathbf{L}}^{\dagger}(t)\mathbf{e}(t)+\hat{\mathbf{L}}^{\dagger}(t)\mathbf{s}(t), \tag{32}$$

where $\hat{\mathbf{L}}^{\dagger}(t)$ denotes an approximation of the pseudo inverse of the image Jacobian matrix $\mathbf{L}(t)$. The applied control law warrants that when the error between the measured and the followed feature parameters is small, the velocity of the object will follow closely the desired velocity generated by the optimization model.

Note that the image Jacobian matrix $\mathbf{L}(t)$ requires information that is not directly available from the image measurements, e.g., partial pose estimation of the object. Therefore, an approximation of the matrix is usually used in VS tasks, with different models for the approximation reported in the literature [9]. The model employed in this work is detailed in Appendix B.

Regarding the stability of the proposed scheme, it is well known that local asymptotic stability is guaranteed in the neighborhood of $\mathbf{e}=\mathbf{0}$ if $\hat{\mathbf{L}}^{\dagger}\mathbf{L}$ is positive definite [9]. Global asymptotic stability cannot be achieved, because $\hat{\mathbf{L}}^{\dagger}$ has a non-zero null space. However, in the neighborhood of the desired pose the control scheme is free of local minima, and the convergence is guaranteed. These properties of the IBVS control scheme render its suitability for tracking tasks, such as the problem at hand. That is, when the features selection and system calibration are reasonably performed so that $\hat{\mathbf{L}}^{\dagger}\mathbf{L}$ is positive definite, then the errors between the current and desired feature parameters will converge to zero along the tracked trajectory. Although the region of local asymptotic stability is difficult to be calculated theoretically, it has been shown that in practice it can be pretty large [9].

For calculations of the robot joint angles, the robot Jacobian matrix is combined with the image Jacobian matrix into a feature Jacobian matrix $\mathbf{J}_s\in\mathbb{R}^{2n\times\xi}$ [9]

$$\mathbf{J}_s(\mathbf{q},t)=\mathbf{L}(t)\begin{bmatrix}\left(\mathbf{R}_c^b\right)^T & \mathbf{0}_{3\times3}\\ \mathbf{0}_{3\times3} & \left(\mathbf{R}_c^b\right)^T\end{bmatrix}\begin{bmatrix}\mathbf{I}_{3\times3} & -S\left(\mathbf{R}_e^b(t)\,\mathbf{P}_o^e\right)\\ \mathbf{0}_{3\times3} & \mathbf{I}_{3\times3}\end{bmatrix}\mathbf{J}\left(\mathbf{q}(t)\right). \tag{33}$$

The joint angles of the robot are updated based on (22) and (32), i.e.:

$$\mathbf{q}(t)=-\lambda\hat{\mathbf{J}}_s^{\dagger}(\mathbf{q},t)\mathbf{e}(t)+\hat{\mathbf{J}}_s^{\dagger}(\mathbf{q},t)\mathbf{s}(t). \tag{34}$$

VII. Validation

The presented approach was first evaluated via simulations in a virtual environment, and afterwards via experiments with a robot in a real-world environment.

The simulations and the experiments were performed in the MATLAB environment on a 3 GHz quad-core CPU with 4 GB of RAM running under Windows 7 OS.

A simulated environment in MATLAB was created using functions from the Visual Servoing Toolbox for MATLAB/Simulink [22] and the Robotics Toolbox for MATLAB [23]. A virtual camera model was created with a visual field of 640×480 pixels. A virtual robot Puma 560 was adopted for execution of the generated reproduction strategy. The goal of the simulations is to evaluate the developed task planning method in the image space using the presented optimization model, rather than validation of the task execution.

An object is moved by a human demonstrator from an initial position to an ending position, while the poses of the object were recorded with an optical marker-based tracker. A total of five demonstrations were collected, displayed in Fig. 3a. The demonstrations involved translatory motions, whereas synthetic rotation of 60 degrees around the *y*-axis was added to all trajectories. It was assumed that the object has six point features (i.e., N = 6) and the 3-dimensional geometric model of the object is readily available.

Note that in the plots that follow the demonstrated trajectories are depicted by thin solid (black) lines. The initial states of the trajectories are indicated by square marks, while the final states are indicated by cross marks. In addition, the reference trajectories are represented by thick dashed (red) lines, the generalized trajectories by SOCO optimization are depicted by thick solid (blue) lines, whereas the robot executed trajectories are differentiated by lines with circle markers (green lines).

The Kalman-smoothed image feature trajectories are displayed in Fig. 3b. The velocities of the demonstrated trajectories are shown in Fig. 3c, and the Kalman-smoothed velocities of the object are shown with the dashed lines in Fig. 3c. For initialization of the Kalman smoothers (Appendix A) the measurement and process noise covariance matrices were set as follows. For each feature point n: $\Sigma_S^n=10\mathbf{I}_{(4\times4)}$, $\Upsilon_S^n=10\mathbf{I}_{(4M\times4M)}$; for the object's pose: $\Sigma_{P_o}=0.1\mathbf{I}_{(6\times6)}$, $\Upsilon_{P_o}=10\mathbf{I}_{(6M\times6M)}$; and, for the object's velocity: $\Sigma_{v_o}=0.1\mathbf{I}_{(6\times6)}$, $\Upsilon_{v_o}=10\mathbf{I}_{(6M\times6M)}$. As noted before, the notation $\mathbf{I}_{(b\times b)}$ refers to an identity matrix of size b.

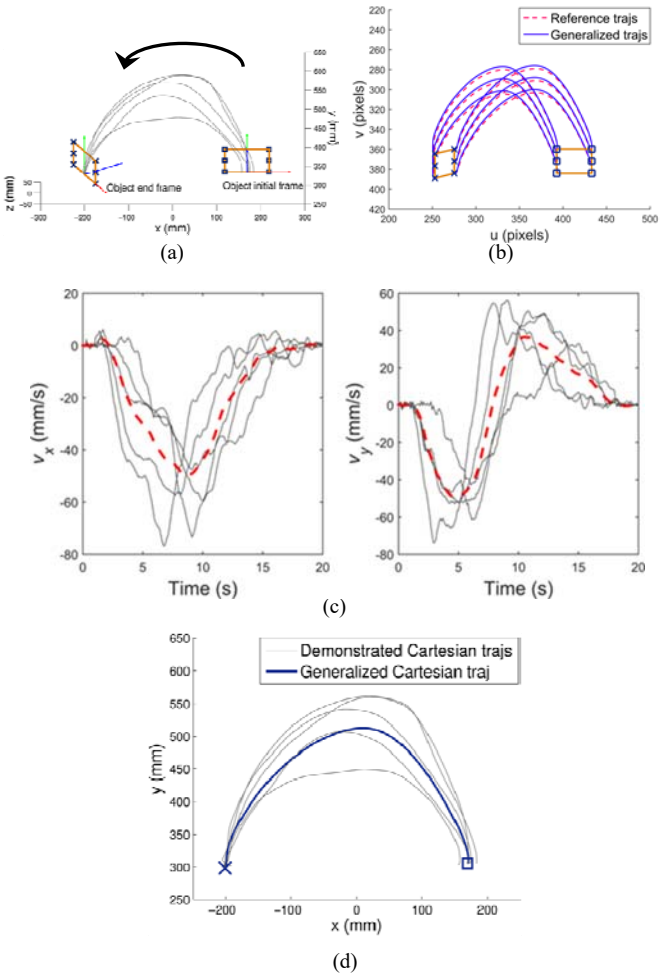


Fig. 3. (a) Demonstrated Cartesian trajectories of the object, showing the features points, and the initial and ending object frames; (b) Reference image feature trajectories from the Kalman smoothing, and the corresponding generalized trajectories from the optimization; (c) Demonstrated and reference linear velocities of the object for *x* and *y* coordinates of the motions; (d) The demonstrated and retrieved generalized object trajectory in the Cartesian space. The initial state and the ending state are depicted with square and cross marks, respectively.

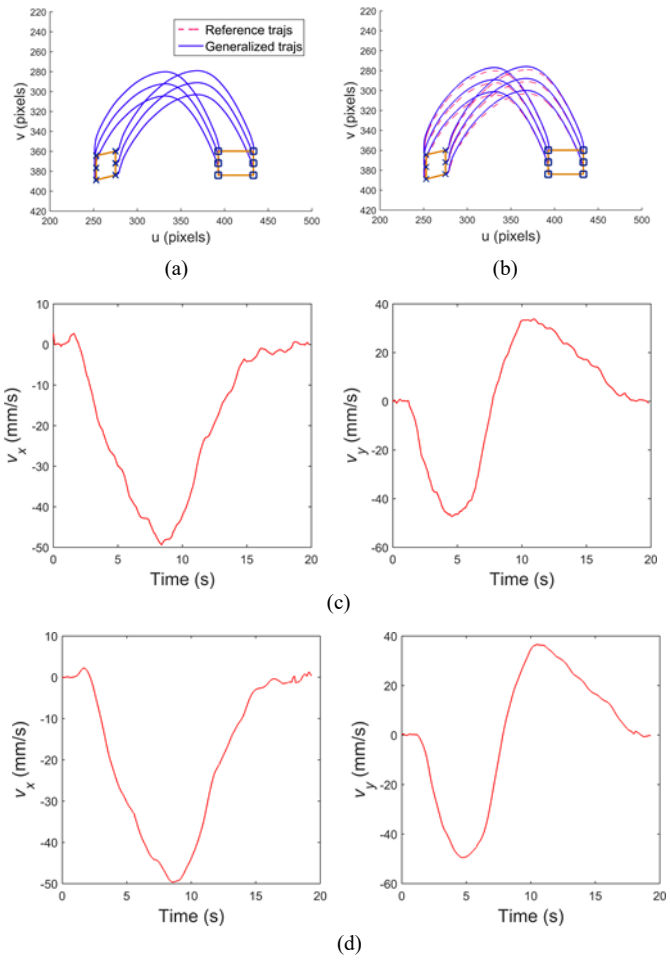


Fig. 4. (a) Reference and generalized image feature trajectories for case 1 of coefficients; (b) Reference and generalized image feature trajectories for case 2 of coefficients; (c) Generalized object velocity for case 1 of coefficients; (d) Generalized object velocity for case 2 of coefficients.

The resulting reference trajectories were then used for initialization of the optimization procedure. The following values for the weighting coefficients were adopted: $\{\alpha_n\}_{n=1}^{6} = 0.1$, $\alpha_v = \alpha_\omega = 0.5$. The weighting scheme assigns higher weight to the importance of following the reference velocities, whereas the model constraints ensure that the generated feature trajectories in the image space are close to the reference trajectories and are within the bounds of the demonstrated task space. The resulting generalized image features trajectories are shown with the solid lines in Fig. 3b. The corresponding generalized Cartesian object trajectory is shown in Fig. 3d. The trajectory is within the bounds of the demonstrated task space, and reaching the final goal state of the task is accomplished. The simulation results of the robot's task execution are not presented, since a more proficient analysis ensues, involving real world experiments.

The impact of the weighting coefficients in (2) on the generalized solution of the optimization is evaluated for two cases of extreme values of the coefficients. The first case is $\{\alpha_n\}_{n=1}^{6} = 1$, $\alpha_v = \alpha_\omega = 0.1$, which places higher weight on the importance of following the reference image feature trajectories. The results are shown in Fig. 4ac. The second case is $\{\alpha_n\}_{n=1}^{6} = 0.01$, $\alpha_v = \alpha_\omega = 1$, and it directs the solution of the optimization problem toward following the reference velocities of the object. The results of this case are shown in Fig. 4bd. For Case 1 the generalized image features trajectories almost coincide with the reference image trajectories, and the dashed lines are hardly visible under the solid lines in Fig. 4a. On the other hand, the obtained object velocities in Fig. 4c are not as smooth as for the second case. In general, it is preferred to apply higher weights on the velocity parameters, so as to relax the requirement to closely follow the reference image feature trajectories.

The approach was next evaluated through experiments with a CRS A255 desktop robot and a Point Grey's 640×480 Firefly MV CMOS camera, both shown in Fig. 5a. The calibration estimated values of the intrinsic camera parameters were: principal point coordinates $u_0 = 296.54$, $v_0 = 266.04$, focal length f = 8.3 mm, and scaling factors $f k_u = 1395.92$, $f k_v = 1397.88$ pixels. An object with 5 coplanar circular

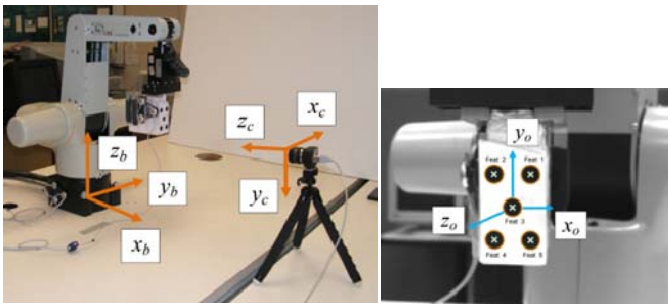


Fig. 5. (a) Experimental setup showing the CRS A255 robot in the home configuration and the camera, with the robot base frame (x_b, y_b, z_b) and camera frame (x_c, y_c, z_c) depicted; (b) Object as seen by the camera in the initial robot configuration. The boundaries and centroids of the features are shown in the figure.

features shown in Fig. 5b was employed for the experiments (i.e., $N = 5$). The coordinates of the features in the object frame were: (12, 22, 0), (-12, 22, 0), (0, 0, 0), (-12, -22, 0), (12, -22, 0) mm. The demonstrations consisted of manipulating the object by a human demonstrator in front of the camera.

The A255 robot endows 5 degrees-of-freedom (DOFs): the arm provides three, and the wrist provides two rotational DOFs, respectively. The fact that the wrist has only two DoFs prevents from exerting arbitrary orientation of the robot's end-point, and subsequently of the manipulated object. Accordingly, we designed the experiments to avoid the shortcomings of the missing rotational DOF. In the first place, one must have in mind that human demonstrated trajectories of an object cannot be completely devoid of a certain rotational DOF, due to the stochastic character of human motion. Therefore, we applied kinesthetic demonstrations for the experiments [24, 25], meaning that while the object is grasped by the robot's gripper, the joint motors are set into passive mode, and the links are manually moved by a demonstrator (see Fig. 6). Note also that the kinesthetic mode of demonstrations does not require solving the correspondence problem between the kinematics of the demonstrator and the robot.

Two sets of experiments were designed and used to evaluate the performance of the proposed approach. The first experiment consisted of moving an object along a trajectory of a pick-and-place task. A sequence of images from the kinesthetic demonstrations is displayed in Fig. 6. Pixel coordinates of the centroids of the dots were considered as image feature parameters. The trajectories of the feature points in the image plane of the camera for one of the demonstrations are presented in Fig. 7a. The frame rate of the camera was set to 30 fps. The total number of demonstrated trajectories M is five.

Tracking of the features in the image sequences was based

Fig. 6. Sequence of images from the kinesthetic demonstrations.

on the 'dot tracker' method in VISP package [26]. Before the manipulation, when the robot was in the initial configuration, the five dots were manually selected in an acquired image. Afterwards, tracking of each feature was achieved by processing the regions of interest with size of 40 pixels, centered at the centroids of the dots in the previous image. The feature extraction involved binarization, thresholding, and centroids calculation of the largest objects with connected pixels. The extracted trajectories ($\mathbf{u}_n^{(m)}(t_k)$ for $n = 1,2,...,5$, $m = 1,2,...,5$, $k = 1,2,...,T_m$) were initially lightly smoothed with a moving average window of 3-points, and linearly scaled to the length of the longest demonstration (which contained 511 frames).

Kalman smoothers were employed to find a smooth average trajectory of each feature point (e.g., $\mathbf{u}_2^{\text{ref}}$ for the feature point 2 is depicted by thick dashed (red) line on Fig. 7b), as well as to find reference velocities of the object $\mathbf{v}_o^{\text{ref}}$ (displayed with thick dashed lines in Fig. 7c). For the Kalman smoothing, the measurement and process noise covariance matrices corresponding to the image feature parameters were $\mathbf{\Sigma}_S^n = 100\mathbf{I}_{(4\times4)}$, $\mathbf{\Upsilon}_S^n = 100\mathbf{I}_{(2M\times2M)}$, respectively. For the object's pose, the corresponding matrices were $\mathbf{\Sigma}_{P_o} = 0.1\mathbf{I}_{(6\times6)}$, $\mathbf{\Upsilon}_{P_o} = 10\mathbf{I}_{(6M\times6M)}$, and, for the object's velocity $\mathbf{\Sigma}_{v_o} = 0.1\mathbf{I}_{(6\times6)}$, $\mathbf{\Upsilon}_{v_o} = 10\mathbf{I}_{(6M\times6M)}$. The initialization based on the above parameters was set with a goal to generate averaged feature trajectories which will follow closely the demonstrated trajectories. On the other hand, the object's positions and velocities were calculated indirectly from the noisy image measurements (by using planar homography transformation), and therefore the noise covariance matrices were selected to impose smoothing to a greater extent (Figs. 7bc).

Afterward, the SOCO optimization model presented in Section V was run, with the reference trajectories obtained from the Kalman smoothing algorithms $\mathbf{s}^{\text{ref}}$ and $\mathbf{v}_o^{\text{ref}}$ used as inputs for the optimization. The discrete sampling period was set equal to $\Delta t_k = 0.033$ seconds, for $k = 1,2,\ ,511$, which corresponds to the camera frame rate. The values for the weighting coefficients were adopted again as $\{\alpha_n\}_{n=1}^5 = 0.1$, $\alpha_v = \alpha_\omega = 0.5$. The overall time expense of the optimization method by using the SeDuMi toolbox was approximately 30 seconds. The output trajectory in pixel coordinates for the feature point 2 is shown in Fig. 7b with a thick solid line, superimposed with the demonstrated trajectories and the reference trajectory. Additionally, the output image trajectories for all five feature points are shown in Fig. 7d, and the corresponding Cartesian trajectory of the object in robot base coordinates is shown in Fig. 7e.

Fig. 7b shows that the reference trajectory obtained by the Kalman smoother (dashed line) does not correspond to the geometric mean of the demonstrations, and at some points it does not belong to the envelope of the demonstrated trajectories. The reason behind it is the temporal variations across the trajectories due to the random distributions of the velocities in the demonstrations, i.e., the trajectories are linearly scaled to an equal length, but they are not temporally

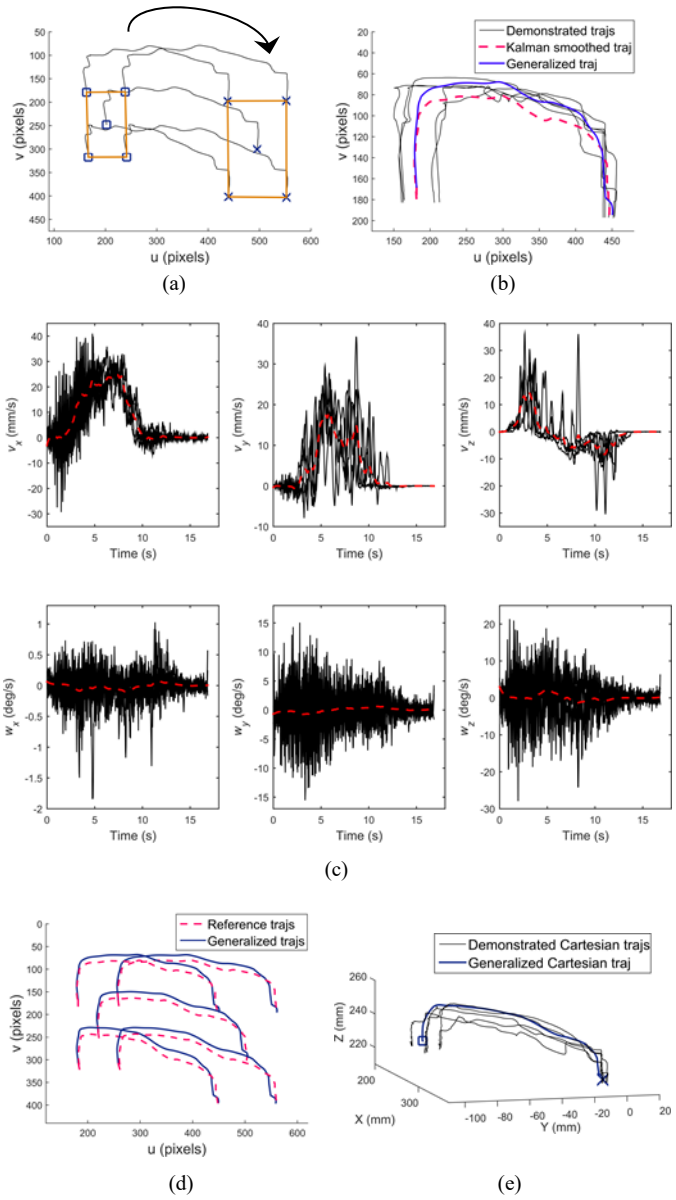


Fig. 7. (a) Image feature trajectories for one of the demonstrations; (b) Demonstrated trajectories, reference trajectory from Kalman smoothing, and the corresponding generalized trajectory for one of the feature points; (c) Demonstrated linear and angular velocities of the object and the reference velocities from Kalman smoothing; (d) Reference image feature trajectories and the SOCO generated image feature trajectories; (e) Demonstrated and the generated Cartesian trajectory of the object in the robot base frame.

scaled. Therefore, the observed features locations at time instant t_k, $\mathbf{u}^{(m)}(t_k)$, correspond to different states (i.e., spatial positions) across the demonstrated set of M trajectories. On the other hand, it can be noticed that the generated trajectory from the SOCO optimization is constrained to lie within the envelope of the demonstrated trajectories. One could note here that if the demonstrated trajectories were temporally scaled (for instance, by using the dynamic time warping algorithm [27], the resulting reference trajectories of the features points will be bounded within the demonstrated envelope, thus it will better represent the task. However, the disadvantage of this approach is the distortion of the velocity profile of the demonstrations caused by temporal warping, which apparently plays an important role since our goal is trajectory planning (as opposed to path planning approaches).

To move the object by the robot along the generalized trajectory from the optimization model, the IBVS tracking control from (34) was employed. Low-level commands to robot's joint motors were sent at 1 millisecond intervals through QuaRC toolbox for open architecture control (from Quanser, Markham, Canada). The resulting trajectories of the object features during the task execution by the robot are shown in Fig. 8a (lines with circle markers). For comparison, the desired trajectories from the optimization process are depicted with thick solid lines. Regarding the selection of the control gain λ, recall that higher gains dictate fast adaptation to the desired trajectory accompanied with reduced accuracy due to overshoots, and vice versa for the lower control gains. Also, setting the control gain too high can destabilize the system. Thus, the parameter λ was set to 0.75 for the first 4/5th of the trajectory length to ensure accurate tracking, and to 0.1 for the last 1/5th of the trajectory to provide accurate positioning at the end of the task. The tracking errors in the image plane are shown in Fig. 8b, for the horizontal (x) and vertical (y) pixel coordinates of the five features. The errors during the trajectory following have moderate values of several pixels, with the errors at the final position settling within 2 pixels. The tracking errors in the Cartesian space are shown in Fig. 8c. The object poses are extracted from the image sequences. The errors at the final object position expressed in robot base frame are (1.05, 0.04, -0.15) mm. The x coordinate corresponds to the depth measurements in the camera frame. As expected, feature depths are more sensitive to image noise, due to the camera projection model. The orientation errors of the object at the final position are (-0.74, 5.2) degrees for the pitch and roll angles. The command values for the linear and angular velocities of the object are shown in Fig. 8d.

The proposed approach is important because it establishes a framework for utilizing the robustness of vision-based control in PbD learning. For instance, consider the trajectory tracking results shown in Fig. 9, which correspond to the same object manipulation task, but this time the reference trajectories from the Kalman smoother were used as desired trajectory for the IBVS tracker. In other words, the optimization procedure was not performed. The results indicate that the tracking was less successful (Fig. 9a), with the steady-state image errors of approximately 5 pixels (Fig. 9b). Moreover, the object position at the end of the task (Fig. 9c) implies divergence, as well as high command velocities at the beginning of the tracking (Fig. 9d). These results reveal the difficulties associated with the trajectories planning in the image space due to non-linearity in the mapping between the image and 3D space. Namely, the desired positions of the five feature points may not correspond to a realizable Cartesian pose of the object, and/or they may impose large velocities of the object in the Cartesian space. In the considered case, the final positions of the image feature parameters are not achievable, i.e., there exist steady-state errors in the image space that the control scheme attempts to minimize, which result in the task failure with regards to the final position of the object (Fig. 9c).

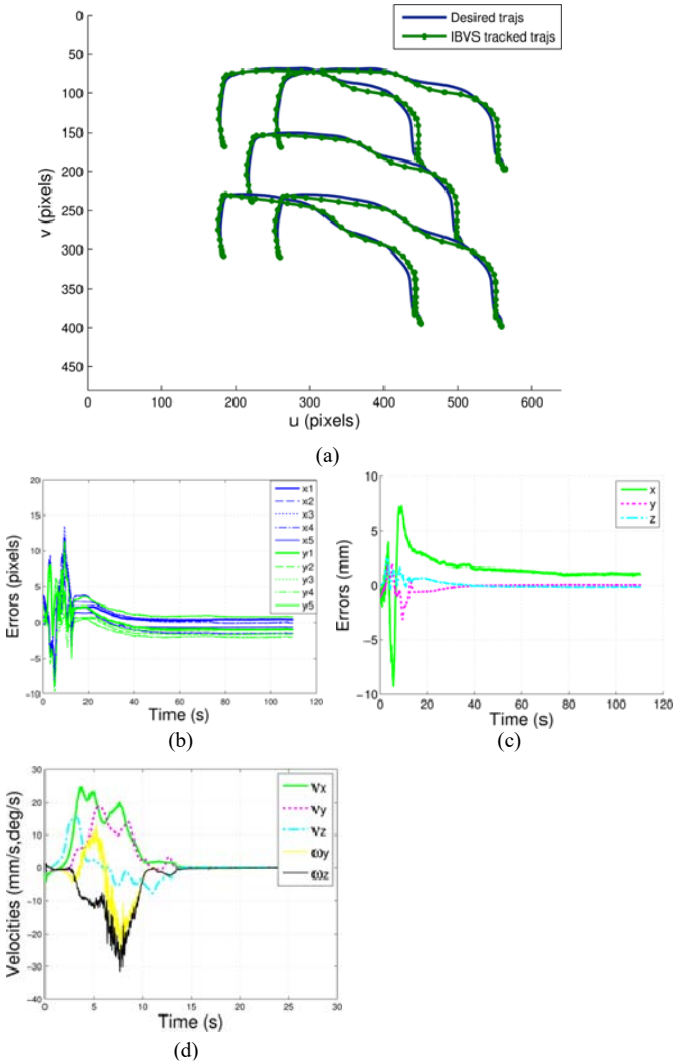


Fig. 8. (a) Desired and robot executed image feature trajectories; (b) Tracking errors for pixel coordinates of the 5 image features in the image space; (c) Tracking errors for *x*, *y* and *z* coordinates in the Cartesian space; (d) Velocities of the object from the IBVS tracker.

Note that if the constraints imposed on the system are too strict, it may be possible that the intersection between the conic space and the affine space in the model (26) is an empty set, i.e., a feasible solution to the optimization does not exist. For instance, the robot joints limits may not allow finding a robot configuration that satisfies the model constraints. By performing kinesthetic demonstrations this scenario can be avoided, since it will ensure that the robot configurations in the demonstrated trajectories are within the dexterous workspace of the robot. In any case, the model parameters should be carefully designed, to provide a basis for existence of solutions to the optimization models.

The advantages of image-based control stemming from the robustness to modeling errors are next evaluated for several types of uncertainties. First, the outcome of the learning from demonstration process is examined under the assumption that the camera is not properly calibrated. The errors in the intrinsic camera parameters ε ranging from 5% to 80%, are introduced as follows: $u_0 = (1+\varepsilon)u_0$, $v_0 = (1+\varepsilon)v_0$, $fk_u = (1+\varepsilon)fk_u$, $fk_v = (1+\varepsilon)fk_v$. Obviously, these calibration

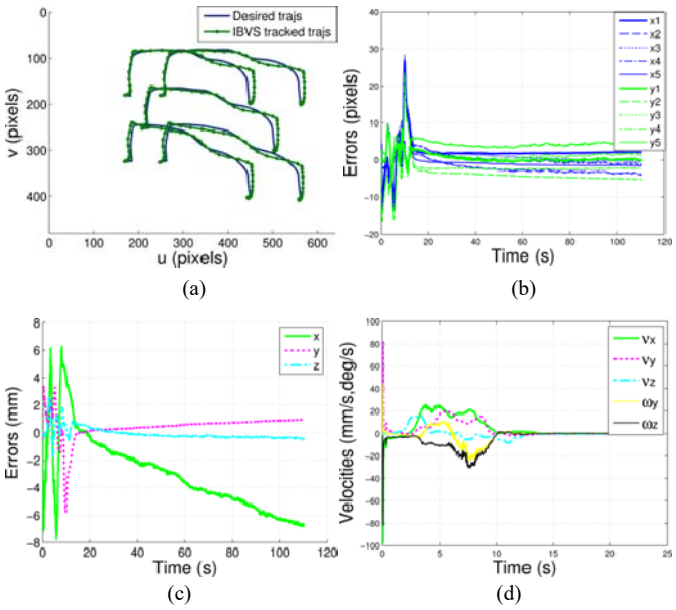


Fig. 9. Task execution without optimization of the trajectories: (a) Desired and robot executed image feature trajectories; (b) Tracking errors for pixel coordinates of the 5 image features in the image space; (c) Tracking errors for *x*, *y* and *z* coordinates in the Cartesian space; (d) Velocities of the object from the IBVS tracker.

errors affect the transformation into image coordinates in (1). In particular, estimation of the object pose from the grabbed images under camera model errors produces erroneous results. In image-based tracking, the control law operates directly over parameters in the image space and compensates for these errors.

The robot performance was evaluated using an optical motion capture system Optotrak® (from NDI, Waterloo, Canada). Optical markers were attached on the object (see Figs. 5, 6), enabling measurements of object's poses with accuracy of 0.1 mm. The following comparison parameters were used for quantifying the deviation between the demonstrated trajectories and the robot executed trajectories:

– steady state-errors of the object's position: $\left|\overline{\xi}_{ss} - \xi_{ss}\right|$, where for each coordinate $\xi \in \{x, y, z\}$, ξ_{ss} denotes the steady state position of robot-executed trajectory, and $\overline{\xi}_{ss} = \left(\sum_{m=1}^{M} \xi_{ss}^{(m)}\right) / M$ denotes the mean steady-state position from the set of *M* demonstrated trajectories;

– root-mean-squared (RMS) deviations for each coordinate $\xi \in \{x, y, z\}$:

$$RMS_{\xi} = \left[\frac{1}{M}\sum_{m=1}^{M}\frac{1}{g}\sum_{k=1}^{g}\left(\xi(t_k) - \xi^{(m)}(t_k)\right)^2\right]^{1/2}; \quad (35)$$

– overall RMS deviations, obtained by summing the deviations for all 3 coordinates:

$$RMS_{total} = \left[\sum_{\xi=x,y,z}\frac{1}{M}\sum_{m=1}^{M}\frac{1}{g}\sum_{k=1}^{g}\left(\xi^2(t_k) - \xi^{(m)}(t_k)\right)^2\right]^{1/2}. \quad (36)$$

For the IBVS tracking the values of the parameters are shown in Table I.

For comparison, the task execution was repeated without vision-based control, that is, the robot uses its motor encoder readings for moving the object. The demonstrated object trajectories in the Cartesian space were extracted from the camera sourced images. Two PbD methods were employed for generating a generalized trajectory: (i) GMM/GMR approach [3] with 8 mixtures of Gaussians used for modeling the demonstrated object positions, and (ii) dynamic movement primitives (DMP) approach [28] with 20 Gaussian kernels used. Operational space velocity controller for the robot's motion was employed in both cases for moving the object along the generalized trajectories. The deviations between the demonstrated trajectories and the robot executed trajectories measured by the optical tracker are given in Table II. The results indicate deteriorated performance, with high tracking errors, as well as errors in the object positioning at the end of the task. The errors originate from: the object pose estimation using erroneous camera parameters, and errors due to robot kinematics.

Robustness to extrinsic camera parameters was also evaluated in a similar fashion. Recall that the extrinsic parameters refer to the pose of the camera with respect to a given frame. The proposed vision-based tracker is very robust to the extrinsic camera errors, since these parameters are hardly used in the control law, with exception of the rotation matrix $\mathbf{R}_c^b$. The obtained results are almost identical for errors from 10% to 80%. The GMM/GMR and DMP approaches were also examined with adding extrinsic camera errors. The camera errors induced erroneous estimation of the object pose from grabbed images, i.e., the object trajectories with respect to robot base were translated into 3D space. However, the performance of both methods was not affected in this case, due to the type of employed robot control scheme. More precisely, the used velocity controller is independent of the robot's Cartesian position, and the object was moved along the desired trajectories without undesired effects by the errors in extrinsic camera parameters. If a position controller was used for moving the robot's gripper (and the object) along the planned trajectories, it would have produced large tracking errors. For comparison, the mean values of the final object pose from the demonstrated trajectories, extracted from camera images with extrinsic errors, are shown in Table III.

Reproduction of learned tasks by using visual servoing can also be implemented by first performing the task planning in the Cartesian space, then projecting several salient features of the target object into the image space, and employing image-based tracker for following of the image features trajectories [12]. This scenario assumes independent planning and execution steps, and ensures robust execution under uncertainties. However, in the case of modeling errors, projections of the features into the image plane can result into trajectories which differ from the planned ones. Fig. 10a shows the projected image plane trajectories from the planning step with thick continuous line. The projected trajectories in cases of intrinsic camera errors of $\varepsilon \in (5, 10, 20)\%$ applied to the four camera intrinsic parameters ($u_0 = (1+\varepsilon)u_0$, $v_0 = (1+\varepsilon)v_0$, $fk_u = (1+\varepsilon)\,fk_u$, $fk_u = (1+\varepsilon)\,fk_u$) are shown overlaid in Fig. 10a. Mapping from the Cartesian into the image space under modeling errors causes deviation from the planned trajectories. Due to the displacement of the principal point coordinates for the case of 20% errors, they are out of the filed-of-view of the camera. Fig. 10b presents another case when moderate 5% errors have been imposed only on the intrinsic parameters corresponding to the focal length scaling factors fk_u and fk_v. The deviation of the projected trajectories from the planned trajectories will result in errors of the robot executed trajectories. On the other hand, the proposed approach uses the camera observed feature trajectories for task planning and the generalized image feature trajectories for task reproduction, thus providing a learning framework robust to camera modeling errors.

The second experiment was designed to evaluate the performance of the proposed approach for trajectories with sharp turns, with the results presented in Fig. 11. The task was repeated four times by a human teacher via kinesthetic demonstrations, where the same learning parameters were used as in the previous experiments. The robot executed

TABLE I
EFFECTS OF INTRINSIC CAMERA PARAMETERS ERRORS ON TRAJECTORY TRACKING WITH IBVS (ERRORS IN MM).

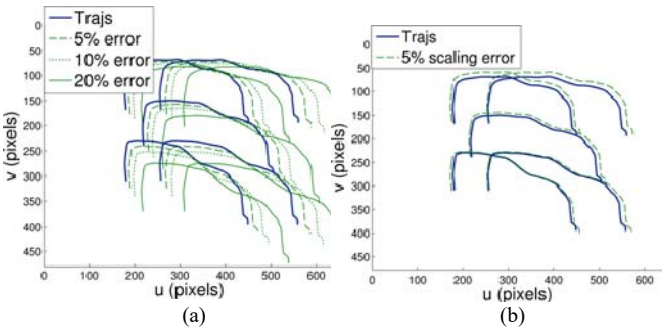
Intrinsic errors	Steady-state errors			RMS errors			Total RMS errors
	x	*y*	*z*	*x*	*y*	*z*	
0 %	8.2	0.2	0.4	11.1	9.3	5.3	15.4
5 %	8.3	0.1	0.5	10.8	8.8	4.8	14.7
10 %	8.3	0.04	0.5	9.4	7.8	3.8	12.8
20 %	8.6	0.02	0.7	10.5	8.3	4.1	13.9
40 %	8.4	0.1	0.6	10.3	7.6	3.5	13.3
80 %	10.6	0.6	0.7	18.7	12.1	6.4	23.2

TABLE II
EFFECTS OF INTRINSIC CAMERA PARAMETERS ERRORS ON TRAJECTORY TRACKING WITHOUT VISION-BASED CONTROL (ERRORS IN MM).

Intrinsic errors	GMM/GMR approach				DMP approach			
	Steady-state errors			RMS total	Steady-state errors			RMS total
	x	*y*	*z*		*x*	*y*	*z*	
0 %	6.5	3.4	17.0	17.7	4.2	1.5	13.3	18.6
5 %	11.7	5.0	18.2	19.1	10.2	3.0	14.8	20.3
10 %	15.9	6.8	20.2	26.5	15.7	5.2	16.4	23.4
20 %	22.3	9.62	22.5	29.9	28.5	9.6	19.6	32.4
40%	33.5	14.3	25.3	36.8	47.1	16.3	25.8	39.8
80 %	46.2	30.4	35.4	43.9	68.6	29.4	36.7	64.9

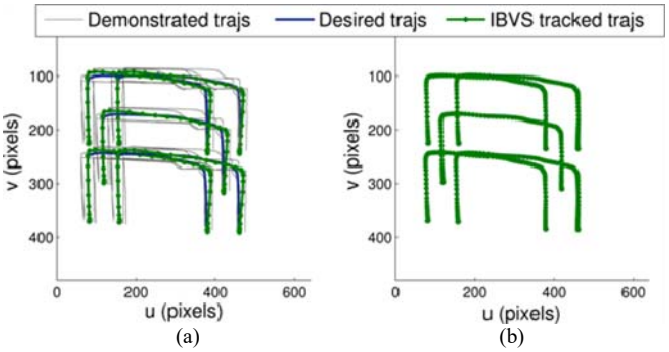
TABLE III
EFFECTS OF EXTRINSIC CAMERA PARAMETERS ERRORS ON OBJECT COORDINATES AT THE END OF THE TASK (MM).

Extrinsic errors	*x*	*y*	*z*
0 %	351.5	-9.1	214.9
10 %	416.6	-12.8	238.5
40 %	611.7	-23.9	309.1
80 %	871.9	-38.9	403.23



(a) (b)

Fig. 10. Projected trajectories of the image feature points with: (a) Errors of 5%, 10% and 20% are introduced for all camera intrinsic parameters; (b) Errors of 5% are introduced for the focal length scaling factors of the camera.



(a) (b)

Fig. 11. (a) Demonstrated trajectories of the image feature points, with the desired and robot executed feature trajectories; (b) Desired and executed trajectories for slowed down trajectories.

trajectory in Fig. 11a indicates overshooting the desired trajectories at the two corners. If the control gain λ was set to lower values, the overshoots can be avoided, but on account of cutting the corners. Fig. 11b shows the task reproduction by the robot, in the case where the generated desired trajectory was interpolated to a trajectory three times longer than the initial trajectory, i.e., this corresponds to slowing down the trajectory three times. In this case, the trajectory following was performed without deviations from the desired trajectory. It can be concluded that the visual tracking performs better for slower trajectories, while fast changing trajectories produce higher tracking errors.

VIII. Discussion

For generating the reference trajectories, instead of Kalman smoothing other machine learning methods can be employed, e.g., Hidden Markov Models [30], GMM/GMR [3], dynamical systems approach [28, 29], etc. However, since the reference trajectories are subjected to correction for visual tracking (through the described optimization procedure), a simple averaging via Kalman smoothing has been adopted here. Although its outputs depend on the initialization, the Kalman smoothing is computationally fast, and can work for trajectories with different shapes complexity [31].

The proposed approach for robot PbD is based on the assumptions of: (i) perception of the demonstrations is performed with vision cameras; and (ii) execution of the learned strategies is conducted using visual feedback from the scene. While most of the PbD works employ electromagnetic, optical, or inertial trackers for capturing the demonstrations, attaching sensors on workpieces and tools is impractical, and unrealistic, for many tasks. Using the robot's 'camera eyes' and possibly other non-intrusive cameras or sensors located on robot's structure, combined with efficient sensor fusion techniques, is a more reasonable solution for the perception step [32, 33]. On the other hand, for reproduction of some tasks by robots it is important to incorporate visual feedback (as well as force feedback [25, 34, 35], which is outside the scope of this work). Without a visual feedback from the scene, the robot would operate blindly, which could potentially lead to unsafe behavior. The above two assumptions are especially relevant for transferring skills to service robots in domestic environments [36, 37].

Among the shortcomings and possible improvements of the approach proposed in this article against the existing PbD methods are: (i) the controlled system is locally stable, however it is difficult to theoretically derive the region of stability; (ii) the implemented convex optimization in the planning phase warrants global convergence within the feasible solutions, however if the constraints are too stringent, a feasible solution might not exist; (iii) the visual tracking can produce large tracking errors for fast changing trajectories; (iv) to derive a generic performance metrics with regards to the optimality of the generated solution for task reproduction is challenging and remains to be done; (v) the demonstrated trajectories are linearly scaled, which can result in poor generalization, especially when the demonstrations vary significantly in duration and velocity; and (vi) the object geometry is assumed to be known, however in a general case it is preferred to consider unknown object geometry.

IX. Conclusion

The article describes a novel approach for trajectory learning from demonstrations, based on performing the steps of perception, planning and execution in the image plane of a vision camera. The proposed framework adopts image-based visual control, due to its robustness to measurement, calibration and modeling errors, when compared to the position-based and hybrid visual servoing methods [8–10].

In order to exploit the advantages of the image-based learning, and to avoid the problems associated with the non-linear mapping between the image space and the Cartesian (task) space, the learning problem was formulated as a convex optimization over a set of variables. The objective of optimization was to generate a feasible set of image feature trajectories and object velocities, while taking into consideration several task constraints. Smoothed averages of the demonstrated trajectories generated by Kalman smoothers were used for initialization of the solutions. The optimization problems were solved off-line at each sampling period, for simultaneously minimizing the sum of distances to the image features parameters and object velocities in the next time instant.

The approach was evaluated experimentally for learning trajectories with different geometry, and for cases of using a coarsely calibrated camera. The results demonstrate that the

designed strategy is tolerant to uncertainties arising from measurements noise and modeling errors. The experiments involving learning with the standard PbD approaches followed by task execution without visual feedback illustrated deteriorated performance, due to errors originating from the robot kinematics and the camera calibration. Although image-based control can be used for task execution as an independent step of the task planning via projection of object features into the image space, the reproduced robot trajectories can differ from the planned ones in the presence of camera modeling errors (Fig. 10).

The formalized integration of the constraints related to the vision-based controller used for task execution into the planning step of PbD generated realizable and locally stable reproduction strategies. The abilities of the robot learner with regards to the limitations of the learning platform were also taken into consideration during the planning step, and formulated as constraints in the optimization procedure.

APPENDIX A: KALMAN SMOOTHER

A system represented by a state transition and observation model equations as follows

$$\begin{cases} \mathbf{x}_k = \mathbf{F}_k \, \mathbf{x}_{k-1} + \mathbf{B}_k \, \mathbf{u}_k + \mathbf{w}_k \\ \mathbf{o}_k = \mathbf{H}_k \, \mathbf{x}_k + \mathbf{v}_k \end{cases}, \tag{A1}$$

is considered, with the notation assumed to be clear from the context of usage for readers familiar with the Kalman filter. The state vector $\mathbf{x}_k$ consists of positions and velocities of the smoothed variables, where the system matrix is defined as

$$\mathbf{F}_k = \begin{bmatrix} 1 & \Delta t_k & & & \\ 0 & 1 & & & \\ & & \ddots & & \\ & & & 1 & \Delta t_k \\ & & & 0 & 1 \end{bmatrix}. \tag{A2}$$

The observation vector $\mathbf{o}_k$ at each time instant k is formed by concatenation of the position measurements from all demonstrated trajectories for the task at hand.

The Kalman smoothing consists of two passes through the observations, where the forward pass applies the regular Kalman filter algorithm, i.e.:

a) Initialization: normal distribution of the process and observation noise variables with zero mean and respective covariances as follow, $\mathbf{w}_k \sim \mathcal{N}(0, \mathbf{\Sigma}_k)$, $\mathbf{v}_k \sim \mathcal{N}(0, \mathbf{\Upsilon}_k)$, initial state $\hat{\mathbf{x}}_0 = \mathbf{x}(0)$, and initial error covariance matrix $\hat{\mathbf{\Pi}}_0 = \hat{\mathbf{\Pi}}(0)$. The error covariance matrix provides a measure of the estimated state, i.e., $\hat{\mathbf{\Pi}}_k = \mathrm{cov}\left(\mathbf{x}_k - \hat{\mathbf{x}}_k\right)$.

b) Prediction:

$$\begin{cases} \hat{\mathbf{x}}_{k|k-1} = \mathbf{F}_k \hat{\mathbf{x}}_{k-1|k-1} + \mathbf{B}_k \mathbf{u}_k \\ \hat{\mathbf{\Pi}}_{k|k-1} = \mathbf{F}_k \hat{\mathbf{\Pi}}_{k-1|k-1} \mathbf{F}_k^T + \mathbf{\Sigma}_k \end{cases}. \tag{A3}$$

c) Update:

$$\begin{cases} \hat{\mathbf{K}}_k = \hat{\mathbf{\Pi}}_{k|k-1} \mathbf{H}_k^T \left(\mathbf{H}_k \hat{\mathbf{\Pi}}_{k|k-1} \mathbf{H}_k^T + \mathbf{\Upsilon}_k \right)^{-1} \\ \hat{\mathbf{\Pi}}_{k|k} = \left(\mathbf{I} - \hat{\mathbf{K}}_k \mathbf{H}_k \right) \hat{\mathbf{\Pi}}_{k|k-1} \\ \hat{\mathbf{x}}_{k|k} = \hat{\mathbf{x}}_{k|k-1} + \hat{\mathbf{K}}_k \left(\mathbf{o}_k - \mathbf{H}_k \hat{\mathbf{x}}_{k|k-1} \right) \end{cases}. \tag{A4}$$

The backward pass performs the smoothing recursively for $k = n-1, n-2, ..., 1$, based on the outcomes of the Kalman filter, through the following steps:

a) Initialization: $\mathbf{x}_n = \hat{\mathbf{x}}_n$, $\mathbf{\Pi}_n = \hat{\mathbf{\Pi}}_n$.

b) Prediction:

$$\begin{cases} \mathbf{x}_{k+1|k} = \mathbf{F}_k \hat{\mathbf{x}}_{k|k} \\ \mathbf{\Pi}_{k+1|k} = \mathbf{F}_k \hat{\mathbf{\Pi}}_{k|k} \mathbf{F}_k^T + \mathbf{\Sigma}_k \end{cases} \tag{A5}$$

b) Update:

$$\begin{cases} \mathbf{K}_k = \mathbf{\Pi}_{k|k} \mathbf{F}_k^T \left(\mathbf{\Pi}_{k+1|k} \right)^{-1} \\ \mathbf{x}_{k|n} = \hat{\mathbf{x}}_{k|k} + \mathbf{K}_k \left(\mathbf{x}_{k+1|n} - \mathbf{x}_{k+1|k} \right) \\ \mathbf{\Pi}_{k|n} = \hat{\mathbf{\Pi}}_{k|k} + \mathbf{K}_k \left(\mathbf{\Pi}_{k+1|n} - \mathbf{\Pi}_{k+1|k} \right) \mathbf{K}_k^T \end{cases}. \tag{A6}$$

In the above equations the variables with overhead caret refer to the filtering process, whereas the notation pertaining to overhead tilde refers to the smoothing process.

APPENDIX B: IMAGE JACOBIAN MATRIX

This part presents the derivation of the image Jacobian matrix for a moving object with respect to a fixed camera. First, consider the coordinates of a point on the rigid object expressed with regard to a static camera frame:

$$\mathbf{P}^c = \mathbf{P}_o^c + \mathbf{R}_o^c \, \mathbf{P}^o. \tag{A7}$$

The linear velocity of the point in camera frame is

$$\mathbf{P}^c = \mathbf{P}_o^c + \mathbf{R}_o^c \, \mathbf{P}^o = \mathbf{P}_o^c + S(\boldsymbol{\omega}_o^c) \, \mathbf{R}_o^c \, \mathbf{P}^o. \tag{A8}$$

From (A7), it follows $\mathbf{R}_o^c \, \mathbf{P}^o = \mathbf{P}^c - \mathbf{P}_o^c$, so (A8) can be rewritten as

$$\mathbf{P}^c = \mathbf{P}_o^c + S(\boldsymbol{\omega}_o^c)\left(\mathbf{P}^c - \mathbf{P}_o^c \right) = \mathbf{P}_o^c + \boldsymbol{\omega}_o^c \times \left(\mathbf{P}^c - \mathbf{P}_o^c \right). \tag{A9}$$

If we denote the point coordinates as $\mathbf{P}^c = \left(X, Y, Z\right)$ and its perspective projection onto the image plane as $x = X / Z$, $y = Y / Z$, from (A9) we have

$$\begin{cases} X = v_x + \omega_y \left(Z - P_{o,x}^c \right) - \omega_z \left(Y - P_{o,y}^c \right) \\ Y = v_y + \omega_z \left(X - P_{o,x}^c \right) - \omega_x \left(Z - P_{o,z}^c \right) \\ Z = v_z + \omega_x \left(Y - P_{o,y}^c \right) - \omega_y \left(X - P_{o,x}^c \right) \end{cases} \tag{A10}$$

where $P_{o,x}^c$, $P_{o,y}^c$, and $P_{o,z}^c$ are the coordinates of the origin of the object frame expressed in the camera frame.

Since $x=\left(XZ-XZ\right)/Z^2$, $y=\left(YZ-YZ\right)/Z^2$, by using (A10), the relationship between the velocities of the feature parameters in image plane and the velocity of the object in the camera frame is found

$$\begin{bmatrix}\dot{x}\\ \dot{y}\end{bmatrix}=\begin{bmatrix}\frac{1}{Z} & 0 & -\frac{x}{Z} & -xy+\frac{xP^c_{o,y}}{Z} & 1+x^2-\frac{P^c_{o,z}}{Z}-\frac{xP^c_{o,x}}{Z} & -y+\frac{P^c_{o,z}}{Z}\\ 0 & \frac{1}{Z} & -\frac{y}{Z} & -1-y^2+\frac{yP^c_{o,y}}{Z}+\frac{P^c_{o,z}}{Z} & xy-\frac{yP^c_{o,x}}{Z} & x-\frac{P^c_{o,x}}{Z}\end{bmatrix}\begin{bmatrix}v_x\\ v_y\\ v_z\\ \omega_x\\ \omega_y\\ \omega_z\end{bmatrix}. \quad \text{(A11)}$$

Note that calculation of the matrix at each time instant requires estimation of the depth of the point Z and the coordinates of the position vector of the object frame with respect to the camera $\mathbf{P}^c_o$. This works employs planar homography transformation for estimation of the depth Z and the position vector $\mathbf{P}^c_o$. While (A11) corresponds to only one feature point, the image Jacobian matrix for several features $\mathbf{L}$ is composed by concatenating the matrices in the form of (A11).

REFERENCES

[1] S. Schaal, A. Ijspeert, and A. Billard, "Computational approaches to motor learning by imitation," *Philosophical Transactions of the Royal Society of London. Biological Sciences*, vol. 358, no. 1431, pp. 537–547, 2003.

[2] S. Calinon, *Robot Programming by Demonstration: A Probabilistic Approach*. Boca Raton, USA: EPFL/CRC Press, 2009.

[3] A.G. Billard, S. Calinon, and R. Dillmann, "Learning from Humans," in *Handbook of Robotics*, B. Siciliano, and O. Khatib, Eds. New York, USA: Springer, pp. 1995–2014, 2016.

[4] B. Argall, S. Chernova, M. Veloso, and B. Browning, "A survey of learning from demonstration," *Robot. Autonom. Syst.*, vol. 57, no. 5, pp. 469–483, 2009.

[5] S. Niekum, S. Osentoski, G. Konidaris, S. Chitta, B. Marthi, and A.G. Barto, "Learning Grounded Finite-State Representations from Unstructured Demonstrations," *Int. Journal Robot. Research*, vol. 34, no. 2, pp. 131–157, 2014.

[6] A. Vakanski, I. Mantegh, A. Irish, and F. Janabi-Sharifi, "Trajectory learning for robot programming by demonstration using Hidden Markov Model and Dynamic Time Warping," *IEEE Trans. Syst., Man, Cybern. - Part B*, vol. 44, no. 4, pp. 1039–1052, 2012.

[7] K.K. Narayanan, L.F. Posada, F. Hoffmann, and T. Bertram, "Acquisition of behavioral dynamics for vision based mobile robot navigation from demonstrations," *IFAC Symp. Mechatronic Systems*, Hangzhou China, pp. 37–44, 2013.

[8] F. Janabi-Sharifi, "Visual servoing: Theory and applications," in *Opto-Mechatronics Systems Handbook*, H. Cho, Ed. Boca Raton, USA: CRC Press, 2002, ch. 15.

[9] F. Chaumette, and S. Hutchinson, "Visual servo control – part I: Basic approaches," *IEEE Robot. Autom. Mag.*, vol. 13, no. 4, pp. 82–90, 2006.

[10] R.T. Fomena, C.P. Quintero, M. Gridseth, and M Jagersand, "Toward Practical Visual Servoing in Robotics," *Int. Conf. Computer Robot Vision*, Saskatchewan, Canada, pp. 303–310, 2013.

[11] H. E. Rauch, F. Tung, and C. T. Striebel, "Maximum likelihood estimates of linear dynamic systems," *J. Amer. Ins. Aeron. Astron.*, vol. 3, no. 8, pp. 1445–1450, 1965.

[12] Y. Mezouar, and F. Chaumette, "Path planning for robust image-based control," *IEEE Trans. Robot. Autom.*, vol. 18, no. 4, pp. 534–549, 2002.

[13] L. Deng, F. Janabi-Sharifi, and W. Wilson, "Hybrid motion control and planning strategies for visual servoing," *IEEE Trans. Ind. Electron.*, vol. 52, no. 4, pp.1024–1040, 2005.

[14] G. Chesi, and T. Shen, "Conferring robustness to path-planning for image-based control," *IEEE Trans. Control Systems Technol.*, vol. 20, no. 4, pp. 950–959, 2012.

[15] A.S. Phung, J. Malzahn, F. Hoffmann, and T. Bertram, "Get out of the way – obstacle avoidance and learning from demonstration for manipulation," *World Congress on IFAC*, Milano, Italy, pp. 11514–11519, 2011.

[16] A.S. Phung, J. Malzahn, F. Hoffmann, and T. Bertram, "Tool centered learning from demonstration for robotic arms with visual feedback," *Int. Conf. Robotics and Bioinformatics*, Guangzhou, China, pp. 1117–1122, 2012.

[17] M. Mitic, and Z. Mijlkovic, "Bio-inspired approach to learning robot motion trajectories and visual control commands," Expert Systems with Applications, vol. 42, pp. 2624–2637, 2015.

[18] O. Faugeras, *Three-dimensional Computer Vision: A Geometric Viewpoint*. Cambridge, USA: MIT Press, 1993.

[19] L. Sciavicco, and B. Siciliano, *Modeling and Control of Robot Manipulators*, 2nd ed. London, UK: Springer, 2005.

[20] S. Boyd, and L. Vandenberghe, *Convex Optimization*. Cambridge, UK: Cambridge University Press, 2004.

[21] J. F. Sturm, *SeDuMi 1.21, a MATLAB Toolbox for Optimization over Symmetric Cones*, 1997 [Online]. Available: http://sedumi.ie. lehigh.edu/

[22] P. Corke, "Robotics, vision and control: Fundamental algorithms in MATLAB," in Springer Tract in Advanced Robotics, vol. 73. (Eds.) Siciliano, B., Khatib, O., and Groen, F., Berlin, Germany: Springer-Verlag, 2011.

[23] E. Cervera, *Visual Servoing Toolbox for MATLAB/Simulink*, 2003 [Online]. Available: http://vstoolbox.sourceforge.net/

[24] M. Hersch, F. Guenter, S. Calinon, and A. Billard, "Dynamical system modulation for robot learning via kinesthetic demonstrations," *IEEE Trans. Robot.*, vol. 24, no. 6, pp. 1463–1467, 2008.

[25] P. Kormushev, D. N. Nenchev, S. Calinon, and D. G. Caldwell, "Upper-body kinesthetic teaching of a free standing humanoid robot," in *Proc. Int. Conf. Robot. Autom.*, Shangai, China, 2011, pp. 3970–3975.

[26] E. Marchand, F. Spindler, and F. Chaumette, "VISP for Visual Servoing: a generic software platform with a wide class of robot control skills," *IEEE Robot. Autom. Mag.*, vol. 12, no. 4, pp. 40–52, 2005.

[27] H. Sakoe, and S. Chiba, "Dynamic programming algorithm optimization for spoken word recognition," *IEEE Trans. Acoustics, Speech, Signal Proces.*, vol. 26, no. 1, pp. 43–49, 1978.

[28] A. J. Ijspeert, J. Nakanishi, and S. Schaal, "Learning attractor landscapes for learning motor primitives," in *Advances in Neural Information Processing Systems 15*, S. Becker, S. Thrun, and K. Obermayer, Eds. Cambridge, USA: MIT Press, 2003, pp. 1547–1554.

[29] S. M. Khansari-Zadeh, and A. Billard, "Learning stable non-linear dynamical systems with Gaussian mixture models," *IEEE Trans. Robot.*, vol. 27, no. 5, pp. 943–957, 2011.

[30] A. Irish, I. Mantegh, and F. Janabi-Sharifi, "A PbD approach for learning pseudo-periodic robot trajectories over curved surfaces," in *Proc. IEEE/ASME Int. Conf. Adv. Integr. Mechatronics*, Montreal, Canada, 2010, pp. 1425–1432.

[31] A. Coates, P. Abbeel, and A. Y. Ng., "Learning for control from multiple demonstrations," in *Proc. Int. Conf. Machine Learning*, Helsinki, Finland, 2008, pp. 144–151.

[32] M. Asada, K. Hosoda, and S. Suzuki, "Vision-based learning and development for emergence of robot behaviors," in *Proc. Int. Symp. Robot. Research*, Hayama, Japan, 1997, pp. 327–338.

[33] M. Asada, Y. Yoshikawa, and K. Hosoda, "Learning by observation without three-dimensional reconstruction," in *Proc. Int. Conf. Intelli. Autono. Syst.*, Venice, Italy, 2000, pp. 555–560.

[34] L. Rozo, P. Jimenez, and C. Torras, "A robot learning from demonstration framework to perform force-based manipulation tasks," *Intelli. Service Robotics*, vol. 6, no. 1, pp. 33–51, 2013.

[35] L. Rozo, D. Bruno, S. Calinon, and D.G. Caldwell, "Learning optimal controllers in human-robot cooperative transportation tasks with position and force constraints," in *Proc. Int. Conf. Intelli. Robots.Syst.*, Hamburg, Germany, pp. 1024–1030, 2015.

[36] A. Kupcsik, D. Hsu, and W.S. Lee, "Learning dynamic robot—to-human handover from human feedback," in *Proc. Int. Symp. Robotics Research*, Genova, Italy, pp. 1–16, 2015.

[37] S. Levine, C. Finn, T. Darrell, and P. Abbeel, "End-to-end training of deep visuomotor policies," *Journal Machine Learning Research*, vol. 17, pp. 1–40, 2016.