On the visual deformation servoing of compliant objects: Uncalibrated control methods and experiments

David Navarro-Alarcon, Yun-hui Liu, Jose Guadalupe Romero and Peng Li

The International Journal of Robotics Research 2014 33: 1462 originally published online 12 June 2014
DOI: 10.1177/0278364914529355

Article

The International Journal of Robotics Research
2014, Vol. 33(11) 1462–1480

DOI: 10.1177/0278364914529355
ijr.sagepub.com

On the visual deformation servoing of compliant objects: Uncalibrated control methods and experiments

David Navarro-Alarcon[1], Yun-hui Liu[1], Jose Guadalupe Romero[2] and Peng Li[1,2]

Abstract

In this paper, we address the active deformation control of compliant objects by robot manipulators. The control of deformations is needed to automate several important tasks, for example, the manipulation of soft tissues, shaping of food materials, or needle insertion. Note that in many of these applications, the object's deformation properties are not known. To cope with this issue, in this paper we present two new visual servoing approaches to explicitly servo-control elastic deformations. The novelty of our kinematic controllers lies in its uncalibrated behavior; our adaptive methods do not require the prior identification of the object's deformation model and the camera's intrinsic/extrinsic parameters. This feature provides a way to automatically control deformations in a model-free manner. The experimental results that we report validate the feasibility of our controllers.

Keywords

Deformation control, visual servoing, adaptive control, robot manipulators, compliant objects, Lyapunov stability

1. Introduction

The robot manipulation of rigid objects has been studied and successfully implemented in several traditional application fields, e.g. pick-and-place and assembly, for more than four decades now—we refer the interested reader to Murray et al. (1994), Okamura et al. (2000), Hokayem and Spong (2006), and Yoshikawa (2010) for comprehensive works on the topic. In recent years, the robotics research community has paid considerable attention to the problem of the automatic manipulation and shaping of deformable objects. In the authors' opinion, this emergent interest in controlling deformations is mainly driven by the now widespread use of surgical robotics (Mallapragada et al., 2011), and the automation of economically important tasks that require control of interactions with soft bodies, e.g. shaping of food materials (Tokumoto et al., 2001), or assembling flexible objects (Park and Mills, 2005).

The objective of the active deformation control is to provide a desired shape or configuration to a compliant object by the interaction of a mechanical system (typically a robot manipulator). One of the principal issues that complicates the automation of these types of tasks is the difficulty of estimating the deformation properties of soft materials. From a control design perspective, this information is needed to coordinate the manipulator's motion with the visual measurements of the object's shape. In other words, we need to know how displacements of the manipulator are "mapped" to deformations of the body.

The deformation properties of a compliant object can be obtained, for example, through traditional offline parameter identification techniques (such as least-squares model fitting). An alternative to this offline and fixed-model approach is to equip model-free controllers with some kind of online adaptation to cope with the unknown elastic properties of an object. Our aim in this paper is precisely to develop uncalibrated methods that do not require the identification of the deformation model.

1.1. Related work

To the best of our knowledge, one of the earliest works to comprehensively address the robotic manipulation of deformable objects is reported in Henrich and Worn (2000).

[1]Department of Mechanical and Automation Engineering, The Chinese University of Hong Kong, Hong Kong SAR, The People's Republic of China
[2]Laboratoire des Signaux et Systemes (L2S), SUPELEC, Gif sur Yvette, France

Corresponding author:
David Navarro-Alarcon, Department of Mechanical and Automation Engineering, The Chinese University of Hong Kong, Rm 112, WMW Mong Engineering Bldg, Shatin NT, Hong Kong SAR, The People's Republic of China.
Email: navarro-alarcon@cuhk.edu.hk

The mathematical modeling and properties estimation of rheological (i.e. food-like) materials is studied in Kimura et al. (2003) and Wang and Hirai (2011). In the past decade, great progress has been obtained in the manipulation of deformable linear objects (see e.g. Sun and Liu, 2001; Wakamatsu and Hirai, 2004; Yue and Henrich, 2005; Saha and Isto, 2007).

As the central topic along this paper, we are mainly interested in the *explicit* deformation servo-control of compliant objects. There are some research works that address this challenging control design problem. For example, a control method (based on a forming process model) to actively shape soft food dough is reported in Tokumoto and Hirai (2002). A method to control the 1-dimensional deformation (i.e. the compression) of a rheological material is reported in Higashimori et al. (2010). To control deformations with rheological materials, these two methods need a-priori knowledge of the visco-elastic properties.

For the case of elastic deformations, researchers have proposed model-based controllers to indirectly position multiple feature *points* (Hirai and Wada, 2000). This indirect positioning problem is also addressed in Torabi et al. (2009); Kinio and Patriciu (2012). The former controllers only consider point deformations of a flexible object/surface. There are some recent studies that model and manipulate deformations in a more general way. For example, Cusumano-Towner et al. (2011) report the results of a two-arm robot that brings fabrics into desired configurations. A numerical study of a model-based controller that deforms a flexible body into a desired contour by several manipulators is reported in Das and Sarkar (2011). Recently, Berenson (2013) developed a method to manipulate soft objects such as fabrics and ropes; this controller relies on a fixed coarse approximation of the object's deformation properties.

Given the nature of the vision-based active shaping problem, it is reasonable to expect that control methods require some information, albeit approximated, of the object's deformation properties. These properties may come in the form of visual/kinematic relations, dynamic models, rheological parameters, rigidity functions, and so on. Note that most works in the literature obtain this information offline (prior the active control). That is, there is very little work about the simultaneous online estimation and control of elastic deformations. Recently, we have developed a model-free controller that estimates on-the-fly the deformations of a flexible object (Navarro-Alarcon et al., 2013). For that, we numerically compute the deformation Jacobian matrix by using the Broyden update rule (Broyden, 1965; Hosoda and Asada, 1994; Jagersand et al., 1997)—a disadvantage of this technique is that it only provides "numerical snapshots" of the unknown deformation matrix.

1.2. Our contribution

To contribute to this economically important problem, in this paper we present two new vision-based control methods to actively shape compliant objects. The methods that we report are uncalibrated since they do not require the identification of the deformation model and the camera's intrinsic/extrinsic parameters. Our first controller estimates in real-time the deformation Jacobian matrix of entirely unstructured elastic environments. Our second and conceptually different approach shows how to combine a minimum of offline information with online adaptation of variable parameters. We report an experimental study to validate the performance of both control methods.

The main difference between previously reported deformation approaches and our control methods is the new uncalibrated adaptive behavior. Note that most existing visual servoing controllers for manipulators, e.g. traditional (Hutchinson et al., 1996; Chaumette and Hutchinson, 2006) or adaptive/uncalibrated methods (Shen et al., 2003; Piepmeier et al., 2004; Liu et al., 2013), do not address active deformation. In this paper, we aim to present energy-motivated *analytical* solutions to the active deformation control problem. This feature contrast with our previous purely numeric algorithm (Navarro-Alarcon et al., 2013). Similar to the online estimator that we present in this paper is our previous method reported in Navarro-Alarcon and Liu (2013). However, note that our earlier approach only considers the affine (linear) camera model, whereas in this new paper we present the complete solution to estimate the deformation Jacobian matrix with perspective cameras. We report an experimental study that compares the performance of these online Jacobian estimators.

1.3. Organization

The rest of this manuscript is organized as follows: in Section 2, we derive the mathematical models. Section 3 states the research problem. In Sections 4 and 5 we present the deformation controllers. Section 6 reports the conducted experimental study. We provide final discussions and future work in Section 7.

2. Modeling

In this section, we present the necessary mathematical modeling of the robot manipulator, the deformable object, and the vision system.

2.1. Notation

Throughout this paper, we employ the following mathematical notation. We denote column vectors and matrices by small bold and capital bold letters, e.g. $\boldsymbol{v} \in \mathbb{R}^{p_1}$, and $\mathbf{M} \in \mathbb{R}^{p_1 \times p_2}$. We denote errors by $\Delta\boldsymbol{v} = \boldsymbol{v} - \boldsymbol{v}^* \in \mathbb{R}^{p_1}$, where $\boldsymbol{v}^*$ represents a desired constant reference. We denote the identity and null matrices by $\mathbf{I}_{p_1 \times p_1} \in \mathbb{R}^{p_1 \times p_1}$ and $\mathbf{0}_{p_1 \times p_2} \in \mathbb{R}^{p_1 \times p_2}$, respectively. We accompany all *implicit* time-varying quantities with a bracket (t), e.g. $\boldsymbol{v}(t)$ and $\mathbf{M}(t)$.

2.2. Robot manipulator

For the deformation problem that we address along this paper, we consider a serial robot manipulator with revolute joints only, that continuously interacts with an elastic environment (Liu and Sun, 2000). We denote the vector of joint positions by $\mathbf{q}(t) \in \mathbb{R}^g$, and the vector of end-effector displacements that produce elastic deformations on the object by $\mathbf{x}(t) \in \mathbb{R}^n$, for $g \geq n$. Note that in practice, $\mathbf{x}(t)$ may not necessarily only represent linear displacements, i.e. in some deformation tasks (especially those involving mechanical gripers) the re-orientation of the end-effector may also produce deformations on a rigidly grasped object.

Remark 1. For ease of presentation, in this paper we formulate the deformation problem with $n = 3$. Note, however, that the same methodology that we report is easily extended to other tasks where $n > 3$.

The differential kinematic equation of this manipulator is given by

$$\dot{\mathbf{x}}(t) = \frac{\partial \mathbf{x}}{\partial \mathbf{q}}(\mathbf{q}(t))\,\dot{\mathbf{q}}(t) \tag{1}$$

where the matrix $\frac{\partial \mathbf{x}}{\partial \mathbf{q}}(\mathbf{q}(t)) \in \mathbb{R}^{3\times g}$ represents the standard Jacobian matrix of the serial kinematic chain (Craig, 1989). For our control purposes, we assume that this Jacobian matrix is exactly known.

We formulate our control methods for kinematically controlled mechanical systems, that is, for manipulators whose control input physically represents joint velocities (Whitney, 1969). We denote the angular velocity input by $\boldsymbol{\omega}(t) \in \mathbb{R}^g$, and without loss of generality assume that $\boldsymbol{\omega}(t) \equiv \dot{\mathbf{q}}(t)$. Note that in real experimental systems the velocity input is usually bounded or subject to saturation (for safety reasons), therefore we model that the components of the joint and end-effector velocity vectors satisfy

$$|\dot{q}_i(t)| \leq c_i, \quad \text{and} \quad |\dot{x}_j(t)| \leq c_j \tag{2}$$

for $c_l > 0 \in \mathbb{R}$ as positive scalars.

2.3. Deformation model

We model the compliant object as a purely elastic body which presents no rheological deformations. To represent the object's configuration, we make use of k deformable feature points conveniently located on the surface of the object. Let the ith point be denoted by the vector $\mathbf{r}_i(t) \in \mathbb{R}^3$.

In our approach, we *locally* model the elastic relation between the end-effector's displacements $\mathbf{x}(t)$ and the coordinates of each point $\mathbf{r}_i(t)$ by the following affine deformation model (Ogden, 1997)

$$\mathbf{r}_i(t) = \mathbf{C}_i\mathbf{x}(t) + \mathbf{o}_i, \tag{3}$$

where the constant matrix $\mathbf{C}_i \in \mathbb{R}^{3\times 3}$ (assumed to be full-rank) and the constant vector $\mathbf{o}_i \in \mathbb{R}^3$ represent the object's deformation parameters. In total, there are $k \times 12$ independent parameters describing the deformation of the points $\mathbf{r}_i(t)$. Note that the affine model (3) represents a local quasi-static approximation of the object's elastic deformation. See Figure 1 for a conceptual representation.

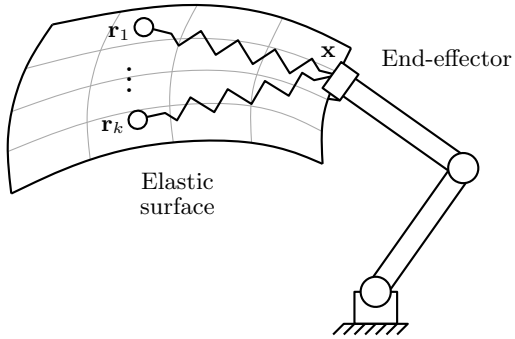


Fig. 1. Conceptual representation of the object's deformation model. We model the Cartesian displacement of the feature points $\mathbf{r}_i(t)$ as a spring interconnection with the manipulator's end-effector $\mathbf{x}(t)$.

2.4. Camera model

To measure the object's deformation, we make use of a fixed camera that provides the visual feedback to the controller in real-time. Let the column vector

$$\mathbf{s}_i(t) = \begin{bmatrix} \mu_i(t) & v_i(t) \end{bmatrix}^{\mathrm{T}} \in \mathbb{R}^2 \tag{4}$$

denote the visual feedback of the feature point $\mathbf{r}_i(t)$; the pixel coordinates $\mu_i(t) \in \mathbb{R}$ and $v_i(t) \in \mathbb{R}$ represent the horizontal and vertical image positions, respectively. To simplify notation, we group all the visual feedback points into a single vector

$$\mathbf{s}(t) = \begin{bmatrix} \mathbf{s}_1^{\mathrm{T}}(t) & \dots & \mathbf{s}_k^{\mathrm{T}}(t) \end{bmatrix}^{\mathrm{T}} \in \mathbb{R}^{2k} \tag{5}$$

We model the projection of the point $\mathbf{r}_i(t)$ onto the image plane with the perspective camera model (Hartley and Zisserman, 2004)

$$\mathbf{s}_i(t) = \frac{1}{z_i(t)}\Big(\mathbf{G}\mathbf{r}_i(t) + \mathbf{b}\Big) \in \mathbb{R}^2 \tag{6}$$

for a constant matrix $\mathbf{G} \in \mathbb{R}^{2\times 3}$ and vector $\mathbf{b} \in \mathbb{R}^2$. See Figure 2 for a conceptual representation of this geometric setup.

The scalar function $z_i(t) \in \mathbb{R}$ represents the *depth* of the point $\mathbf{r}_i(t)$ from the camera center in the direction of the ray normal to the image plane. This scalar distance satisfies

$$z_i(t) = \mathbf{g}_3^{\mathrm{T}}\mathbf{r}_i(t) + b \tag{7}$$

for a constant vector $\mathbf{g}_3 \in \mathbb{R}^3$ and scalar $b \in \mathbb{R}$. It is well-known that for any imaged points $\mathbf{s}_i(t)$, the distance $z_i(t) > 0$ is always positive.

We can group all the camera's constant terms into a single 3×4 matrix

$$\mathbf{M} = \begin{bmatrix} \mathbf{G} & \mathbf{b} \\ \mathbf{g}_3^{\mathrm{T}} & b \end{bmatrix} \in \mathbb{R}^{3\times 4} \tag{8}$$

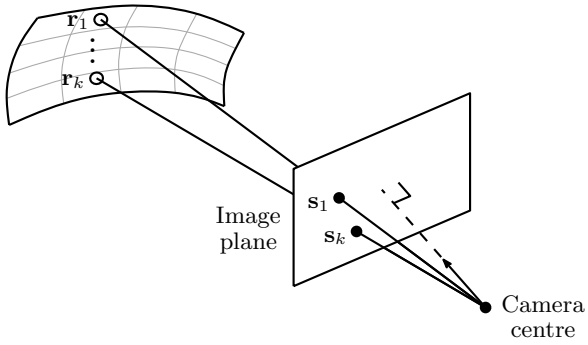


Fig. 2. Perspective projection of the feature points onto the image plane of a fixed camera. The vectors $\mathbf{r}_i(t)$ and $\mathbf{s}_i(t)$ denote the point's Cartesian and image coordinates, respectively.

which is commonly referred as the perspective projection matrix. The matrix $\mathbf{M}$ is constructed with the intrinsic (i.e. the internal calibration) and extrinsic (i.e. the physical setup) parameters of the camera and has always rank 3. With this matrix we can express the projection (6) as a linear mapping between homogeneous vectors

$$z_i(t)\begin{bmatrix}\mathbf{s}_i(t)\\1\end{bmatrix}=\mathbf{M}\begin{bmatrix}\mathbf{r}_i(t)\\1\end{bmatrix} \tag{9}$$

By computing the time-derivative of the projection model (6), we obtain the differential relation between the end-effector motion $\dot{\mathbf{x}}(t)$ and the optical flow $\dot{\mathbf{s}}_i(t)$

$$\begin{aligned}\dot{\mathbf{s}}_i(t)&=\frac{1}{z_i(t)}\Big(\mathbf{G}\dot{\mathbf{r}}_i(t)-\mathbf{s}_i(t)\dot{z}_i(t)\Big)\\&=\frac{1}{z_i(t)}\Big(\mathbf{G}\mathbf{C}_i-\mathbf{s}_i(t)\,\mathbf{g}_3^{\mathrm{T}}\mathbf{C}_i\Big)\dot{\mathbf{x}}(t)\end{aligned} \tag{10}$$

2.5. Deformation-projection parameters

Using the deformation model (3), we construct the *deformation-projection* matrix of the ith point as

$$\mathbf{L}_i\cong\kappa\begin{bmatrix}\mathbf{G}\mathbf{C}_i & \mathbf{a}_i\\ \mathbf{g}_3^{\mathrm{T}}\mathbf{C}_i & a_i\end{bmatrix}\in\mathbb{R}^{3\times4} \tag{11}$$

for $\kappa\neq0\in\mathbb{R}$ as an arbitrary non-zero scalar, and with constant terms $\mathbf{a}_i$ and a_i defined as

$$\mathbf{a}_i=\mathbf{G}\mathbf{o}_i+\mathbf{b}\in\mathbb{R}^2,\qquad a_i=\mathbf{g}_3^{\mathrm{T}}\mathbf{o}_i+b\in\mathbb{R} \tag{12}$$

The relation between the end-effector position and the visual measurements of the ith deformable point can be expressed in terms of the following homogeneous vectors:

$$z_i(t)\begin{bmatrix}\mathbf{s}_i(t)\\1\end{bmatrix}=\mathbf{L}_i\begin{bmatrix}\mathbf{x}(t)\\1\end{bmatrix} \tag{13}$$

In (11), we use the symbol $\cong$ to express the fact that the rank-3 matrix $\mathbf{L}_i$ can only be determined up to a scaling factor (Forsyth and Ponce, 2002; Hartley and Zisserman, 2004).[1] We can check this property in the projection model (6), where the division by the depth clearly cancels κ

$$\frac{\kappa\,(\mathbf{G}\mathbf{C}_i\mathbf{x}(t)+\mathbf{a}_i)}{\kappa\,(\mathbf{g}_3^{\mathrm{T}}\mathbf{C}_i\mathbf{x}(t)+a_i)} \tag{14}$$

The arbitrary scaling factor κ removes 1 degree of freedom from $\mathbf{L}_i$. Thus, this 3×4 matrix only has 11 independent elements. In our method, we fix the scalar a_i with a known value (e.g. 1) and list the independent elements of the k matrices $\mathbf{L}_i$ in the vector of parameters

$$\begin{aligned}\boldsymbol{\theta}=\big[&\mathbf{g}_1^{\mathrm{T}}\mathbf{C}_1\ \ \mathbf{g}_2^{\mathrm{T}}\mathbf{C}_1\ \ \mathbf{g}_3^{\mathrm{T}}\mathbf{C}_1\ \ \mathbf{a}_1^{\mathrm{T}}\ \ \cdots\\&\mathbf{g}_1^{\mathrm{T}}\mathbf{C}_k\ \ \mathbf{g}_2^{\mathrm{T}}\mathbf{C}_k\ \ \mathbf{g}_3^{\mathrm{T}}\mathbf{C}_k\ \ \mathbf{a}_k^{\mathrm{T}}\big]^{\mathrm{T}}\in\mathbb{R}^l\end{aligned} \tag{15}$$

where $l=k\times11$, and the vectors $\mathbf{g}_1\in\mathbb{R}^3$ and $\mathbf{g}_2\in\mathbb{R}^3$ represent the rows of the matrix

$$\mathbf{G}=\begin{bmatrix}\mathbf{g}_1^{\mathrm{T}}\\ \mathbf{g}_2^{\mathrm{T}}\end{bmatrix} \tag{16}$$

3. Problem statement

To avoid the open-loop deformation response that arises with traditional robot-centered approaches, e.g. Mills and Ing (1996); Sun et al. (1999), in this section we use visual feedback to construct a quantitative measurement of the object's deformation. To this end, we introduce the *deformation feature vector*

$$\mathbf{y}(t)=\mathbf{y}(\mathbf{s}(t))\in\mathbb{R}^h \tag{17}$$

where $h\leq3$. The coordinates of the *smooth* (and possibly nonlinear) function $\mathbf{y}(t)$ can be composed of, e.g., point and line features (Wang et al., 2008), areas of polygons (Liu et al., 2013), angles and curvatures (Navarro-Alarcon et al., 2013), or image moments (Chaumette, 2004), to name a few cases.

By differentiating (17) with respect to time, we obtain the differential kinematic expression

$$\dot{\mathbf{y}}(t)=\mathbf{J}(\mathbf{x}(t))\,\dot{\mathbf{x}}(t) \tag{18}$$

with a matrix $\mathbf{J}(\mathbf{x}(t))\in\mathbb{R}^{h\times3}$ defined by

$$\mathbf{J}(\mathbf{x}(t))=\frac{\partial\mathbf{y}}{\partial\mathbf{s}}(\mathbf{s}(t))\,\frac{\partial\mathbf{s}}{\partial\mathbf{x}}(\mathbf{x}(t)) \tag{19}$$

We call $\mathbf{J}(\mathbf{x}(t))$ the *deformation Jacobian matrix* since it relates the motion of the manipulator with the deformation flow. Note that in order to exactly compute the matrix $\mathbf{J}(\mathbf{x}(t))$, we must know the vector of parameters $\boldsymbol{\theta}$.

For ease of presentation, we make a change in the control input variable. Along this paper, we design the deformation control law in terms of the end-effector velocities, that is

$$\mathbf{v}(t)=\dot{\mathbf{x}}(t)\in\mathbb{R}^3 \tag{20}$$

represents the new control variable, which we assume to be a smooth input function for all t.

To clarify the problem that we address in this work, consider the following *uncalibrated* scenario:

- The robot manipulator interacts with an elastic object whose deformation properties are not known.
- The intrinsic and extrinsic parameters of the fixed camera that provides the visual feedback are not known.

Problem. Given a desired and reachable deformation feature vector $\mathbf{y}^* \in \mathbb{R}^h$, design an uncalibrated velocity controller $\mathbf{v}(t)$ which asymptotically minimizes the error $\Delta\mathbf{y}(t) = \mathbf{y}(t) - \mathbf{y}^* \in \mathbb{R}^h$.

4. Deformation controller with online estimation of J(x)

In this section we present a novel control method that online estimates the deformation Jacobian matrix of the unknown elastic object. We formulate this control method similar to most standard vision-based controllers, that is, we first design a desired motion command in deformation coordinates and then map its control action to end-effector velocities using the estimated Jacobian matrix. The originality of our method lies in the proposed online estimation algorithm and the dynamic-state feedback design of the velocity control law.

4.1. Flow estimation error

For ease of presentation, let us first introduce the matrix

$$\mathbf{T}(t) = \begin{bmatrix} \mathbf{G}\mathbf{C}_1 - \mathbf{s}_1(t)\,\mathbf{g}_3^{\mathrm{T}}\mathbf{C}_1 \\ \vdots \\ \mathbf{G}\mathbf{C}_k - \mathbf{s}_k(t)\,\mathbf{g}_3^{\mathrm{T}}\mathbf{C}_k \end{bmatrix} \in \mathbb{R}^{2k\times 3} \tag{21}$$

which comes from stacking up the k depth independent Jacobian matrices in (10), and the matrix

$$\mathbf{Z}(t) = \mathrm{diag}(\mathbf{I}_{2\times 2}z_1(t), \ldots, \mathbf{I}_{2\times 2}z_k(t)) \in \mathbb{R}^{2k\times 2k} \tag{22}$$

which simply represents a block diagonal matrix constructed with the depth of the feature points.

Now, let us assume that at the time instant t we have an estimation of the vector of parameters $\boldsymbol{\theta}$ as defined in (15); we denote this vector of variable parameters by $\widehat{\boldsymbol{\theta}}(t) \in \mathbb{R}^l$. Using this vector, we compute in real-time the *flow estimation error*

$$\mathbf{e}(t) = \widehat{\mathbf{T}}(t)\,\dot{\mathbf{x}}(t) - \widehat{\mathbf{Z}}(t)\,\dot{\mathbf{s}}(t) \in \mathbb{R}^{2k} \tag{23}$$

where $\widehat{\mathbf{T}}(t) \in \mathbb{R}^{2k\times 3}$ and $\widehat{\mathbf{Z}}(t) \in \mathbb{R}^{2k\times 2k}$ represent estimations of the matrices $\mathbf{T}(t)$ and $\mathbf{Z}(t)$, respectively. These estimated matrices are computed with the vector of variable parameters $\widehat{\boldsymbol{\theta}}(t)$ and the arbitrary constants a_i.

The error (23) physically represents the scaled difference between the estimated and measured optical flows. We scale this error by the estimated depth matrix $\widehat{\mathbf{Z}}(t)$ in order to obtain an expression that is linear with respect to the parameters' error $\Delta\boldsymbol{\theta}(t) = \widehat{\boldsymbol{\theta}}(t) - \boldsymbol{\theta} \in \mathbb{R}^l$. To see this, we add the term $\mathbf{Z}(t)\,\dot{\mathbf{s}}(t) - \mathbf{T}(t)\,\dot{\mathbf{x}}(t) = \mathbf{0}_{2k\times 1}$ to (23) to obtain

$$\begin{aligned}\mathbf{e}(t) &= \left(\widehat{\mathbf{T}}(t) - \mathbf{T}(t)\right)\dot{\mathbf{x}}(t) - \left(\widehat{\mathbf{Z}}(t) - \mathbf{Z}(t)\right)\dot{\mathbf{s}}(t) \\ &= \mathbf{Q}(\mathbf{x}(t), \dot{\mathbf{x}}(t), \mathbf{s}(t), \dot{\mathbf{s}}(t))\,\Delta\boldsymbol{\theta}(t)\end{aligned} \tag{24}$$

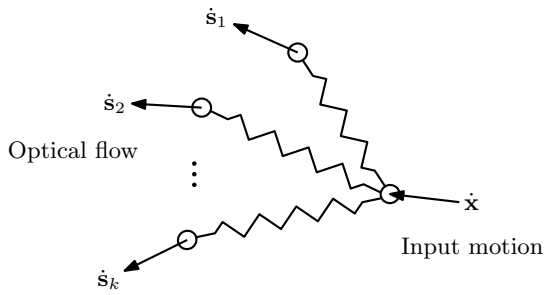


Fig. 3. Conceptual representation of the manipulator motion–optical flow mapping $\dot{\mathbf{x}}(t) \mapsto \dot{\mathbf{s}}(t)$. The purpose of our estimator is to dynamically identify this unknown deformation mapping.

where $\mathbf{Q}(t) \in \mathbb{R}^{2k\times l}$ represents a known regression matrix which does not depend on $\boldsymbol{\theta}$. We compute this matrix by

$$\mathbf{Q}(t) = \begin{bmatrix} \mathbf{F}_1(t) & \mathbf{0}_{2\times 11} & \cdots & \mathbf{0}_{2\times 11} \\ \mathbf{0}_{2\times 11} & \mathbf{F}_2(t) & \cdots & \mathbf{0}_{2\times 11} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{0}_{2\times 11} & \mathbf{0}_{2\times 11} & \cdots & \mathbf{F}_k(t) \end{bmatrix} \tag{25}$$

where the submatrix $\mathbf{F}_i(t) \in \mathbb{R}^{2\times 11}$ is defined for each point $\mathbf{s}_i(t)$ as

$$\begin{aligned}&\mathbf{F}_i(\mathbf{x}(t), \dot{\mathbf{x}}(t), \mathbf{s}_i(t), \dot{\mathbf{s}}_i(t)) = \\ &\quad\begin{bmatrix} \dot{\mathbf{x}}^{\mathrm{T}}(t) & \mathbf{0}_{1\times 3} & -\frac{\mathrm{d}}{\mathrm{d}t}(\mu_i(t)\,\mathbf{x}^{\mathrm{T}}(t)) & \mathbf{0}_{1\times 2} \\ \mathbf{0}_{1\times 3} & \dot{\mathbf{x}}^{\mathrm{T}}(t) & -\frac{\mathrm{d}}{\mathrm{d}t}(\nu_i(t)\,\mathbf{x}^{\mathrm{T}}(t)) & \mathbf{0}_{1\times 2} \end{bmatrix}\end{aligned} \tag{26}$$

4.2. Online estimator

In order to minimize the deformation error $\Delta\mathbf{y}(t)$, we must first estimate how the motion of the manipulator is transformed into deformations of the object (see Figure 3 for a conceptual representation). To this end, in the following proposition we present and analyze the stability of a flow-based algorithm that iteratively estimates this unknown deformation mapping.

Proposition 4.1. *For slow and smooth motion of the manipulator, the parameter update rule*

$$\dot{\widehat{\boldsymbol{\theta}}}(t) = -\boldsymbol{\Gamma}_1\mathbf{Q}^{\mathrm{T}}(t)\,\mathbf{K}_1\mathbf{e}(t) \tag{27}$$

with $\boldsymbol{\Gamma}_1 > 0 \in \mathbb{R}^{l\times l}$ and $\mathbf{K}_1 > 0 \in \mathbb{R}^{2k\times 2k}$ as symmetric tuning matrices, guarantees the following two conditions:

1. *A numerically stable computation of the variable parameters $\widehat{\boldsymbol{\theta}}(t)$.*
2. *The asymptotic minimization of the flow estimation error $\mathbf{e}(t)$.*

Proof. We present a sketch of this proof.

1. By substituting (24) into the update rule (27) we obtain

$$\dot{\widehat{\boldsymbol{\theta}}}(t) = -\boldsymbol{\Gamma}_1\mathbf{Q}^{\mathrm{T}}(t)\,\mathbf{K}_1\mathbf{Q}(t)\,\Delta\boldsymbol{\theta}(t) \tag{28}$$

To analyze the stability of this update rule, we introduce the energy-like function

$$\mathcal{L}(t) = \frac{1}{2}\Delta\boldsymbol{\theta}^{\mathrm{T}}(t)\,\boldsymbol{\Gamma}_1^{-1}\Delta\boldsymbol{\theta}(t) \in \mathbb{R} \tag{29}$$

whose time derivative evaluated along trajectories of (28) satisfies

$$\begin{aligned}\dot{\mathcal{L}}(t) &= -\Delta\boldsymbol{\theta}^{\mathrm{T}}(t)\,\mathbf{Q}^{\mathrm{T}}(t)\,\mathbf{K}_1\mathbf{Q}(t)\,\Delta\boldsymbol{\theta}(t),\\ &= -\mathbf{e}^{\mathrm{T}}(t)\,\mathbf{K}_1\mathbf{e}(t)\end{aligned} \tag{30}$$

This proves that the numerical computation of the variable parameters is stable, i.e. $\widehat{\boldsymbol{\theta}}(t)$ is bounded (Slotine and Li, 1991).

2. Since we consider slow (namely saturated) and smooth (i.e. differentiable) motion of the manipulator, then $\dot{\mathbf{x}}(t)$, $\dot{\mathbf{s}}(t)$, $\ddot{\mathbf{x}}(t)$, and $\ddot{\mathbf{s}}(t)$ are all bounded. After simple computations we can show that

$$\ddot{\mathcal{L}}(t) = -2\mathbf{e}^{\mathrm{T}}(t)\,\mathbf{K}_1\dot{\mathbf{e}}(t), \tag{31}$$

is also bounded, and hence the expression (30) is uniformly continuous. From the Barbalat's lemma (Slotine and Li, 1991), we prove the asymptotic minimization of the flow estimation error, i.e. $\mathbf{e}(t) \to \mathbf{0}_{2k\times 1}$ as $t \to \infty$. Note that the dissipation-like matrix $\mathbf{Q}^{\mathrm{T}}(t)\,\mathbf{K}_1\mathbf{Q}(t)$ is not full rank. This condition simply means that the update rule (27) can not guarantee the estimation of the true parameters. ■

In our uncalibrated method, we compute the deformation Jacobian matrix in real-time as follows:

$$\widehat{\mathbf{J}}(t) = \frac{\partial \mathbf{y}}{\partial \mathbf{s}}(\mathbf{s}(t))\,\widehat{\mathbf{Z}}^{-1}(t)\,\widehat{\mathbf{T}}(t) \in \mathbb{R}^{h\times n}, \quad \forall \widehat{z}_i(t) \neq 0 \tag{32}$$

Remark 2. The objective of this method is not to identify the true camera/deformation models, but to implement an online algorithm that continuously satisfies the optical flow $\dot{\mathbf{s}}(t) = \widehat{\mathbf{Z}}^{-1}(t)\,\widehat{\mathbf{T}}(t)\,\dot{\mathbf{x}}(t)$—see (Hosoda and Asada, 1994) for a similar discussion. Note that since the function of deformation features (17) is exactly known, the algorithm allows us to compute on-the-fly and with no prior model calibration a Jacobian matrix that reconstructs the output deformation flow

$$\dot{\mathbf{y}}(t) = \widehat{\mathbf{J}}(t)\,\dot{\mathbf{x}}(t) \tag{33}$$

Remark 3. The affine expression (3) locally models the quasi-static relation between the deformable points $\mathbf{r}_i(t)$ and the end-effector displacements $\mathbf{x}(t)$. For wider deformations, we can still use our online estimator since it continuously adjusts (or fits) the variable parameters with respect to the visual deformation measurements; note that *slow motion* is recommended with this method.

4.3. Velocity control input

To enforce a desired behavior to the deformation plant (33), we design the following *dynamic-state feedback* velocity control input

$$\mathbf{v}(t) = \widehat{\mathbf{J}}^{+}(t)\,\mathbf{p}(t) \tag{34}$$

where the matrix $\widehat{\mathbf{J}}^{+}(t) \in \mathbb{R}^{3\times h}$ represents the Moore–Penrose pseudoinverse which is defined as

$$\widehat{\mathbf{J}}^{+}(t) = \widehat{\mathbf{J}}^{\mathrm{T}}(t)\left(\widehat{\mathbf{J}}(t)\,\widehat{\mathbf{J}}^{\mathrm{T}}(t)\right)^{-1} \tag{35}$$

The vector $\mathbf{p}(t) \in \mathbb{R}^h$ denotes a numerical state variable which we compute by

$$\dot{\mathbf{p}}(t) = -\frac{\partial \mathcal{U}}{\partial \mathbf{y}}^{\mathrm{T}}(\Delta\mathbf{y}(t)) - \mathbf{D}\mathbf{p}(t) \tag{36}$$

for $\mathcal{U}(\Delta\mathbf{y}(t)) \in \mathbb{R}$ as a positive-definite potential function with a unique equilibrium at $\Delta\mathbf{y}(t) = \mathbf{0}_{h\times 1}$, and $\mathbf{D} > 0 \in \mathbb{R}^{h\times h}$ as a symmetric damping-like matrix.

The deformation control input (34) is designed in a "similar" way to most standard visual servo velocity controllers (Chaumette and Hutchinson, 2006), and in general, similar to the classical kinematic control design for robot manipulators (see e.g. Whitney, 1969; Siciliano, 1990; Nakamura, 1991). With this control approach, we use the inverse of the online estimated Jacobian matrix to map a desired deformation control action to end-effector velocities. The originality of the control design that we present in this section, comes with the use of the dynamic-state feedback action $\mathbf{p}(t)$ to command the desired motion to the manipulator—this contrast with most traditional visual servo controllers that only rely on static-state feedback controls.

The main purpose of introducing the numerical state variable $\mathbf{p}(t)$ (whose physical interpretation corresponds to visual momenta) is to enforce in closed-loop a dynamical system that resembles a classical Hamiltonian system (Marsden and Ratiu, 1994). In other words, by commanding the deformation control action through $\mathbf{p}(t)$, we can design the desired behavior in terms of the familiar "physical" concepts of energy storage and energy dissipation (Hogan, 1985; Ortega et al., 2001).

In the following proposition, we analyze the stability properties of the dynamic-state feedback velocity control law (34).

Proposition 4.2. *Consider that at the time instant t the computed Jacobian matrix* $\widehat{\mathbf{J}}(t)$ *exactly satisfies the kinematic expression* (33)*. For this situation, the control input* (34) *enforces an energetically passive closed-loop system which asymptotically minimizes the deformation error* $\Delta\mathbf{y}(t)$*.*

Proof. Direct substitution of the controller (34) into the deformation plant (33) with $\mathbf{p}(t)$ computed as in (36) enforces

$$\begin{bmatrix}\dot{\mathbf{y}}(t)\\ \dot{\mathbf{p}}(t)\end{bmatrix} = \begin{bmatrix}\mathbf{0}_{h\times h} & \mathbf{I}_{h\times h}\\ -\mathbf{I}_{h\times h} & -\mathbf{D}\end{bmatrix}\begin{bmatrix}\frac{\partial \mathcal{H}}{\partial \mathbf{y}}^{\mathrm{T}}(t)\\ \frac{\partial \mathcal{H}}{\partial \mathbf{p}}^{\mathrm{T}}(t)\end{bmatrix} \tag{37}$$

with a positive-definite (Hamiltonian-like) energy function $\mathcal{H}(\Delta\mathbf{y}(t),\mathbf{p}(t)) \in \mathbb{R}$ defined as

$$\mathcal{H}(\Delta\mathbf{y}(t),\mathbf{p}(t)) = \mathcal{U}(\Delta\mathbf{y}(t)) + \frac{1}{2}\mathbf{p}^{\mathrm{T}}(t)\,\mathbf{p}(t) \tag{38}$$

The scalar functional $\mathcal{H}(\Delta\mathbf{y}(t),\mathbf{p}(t))$ qualifies as a Lyapunov function for (37) since its time-derivative satisfies

$$\dot{\mathcal{H}}(\Delta\mathbf{y}(t),\mathbf{p}(t)) = -\mathbf{p}^{\mathrm{T}}(t)\,\mathbf{D}\mathbf{p}(t) \leq 0 \tag{39}$$

With this expression we prove that the closed-loop system is energetically passive (Arimoto, 1996), i.e. $\mathcal{H}(t_1) \geq \mathcal{H}(t_2)$, $\forall t_1 \leq t_2$. To prove asymptotic stability of the deformation error, we analyze the equilibria of system (37), i.e.

$$\mathbf{0}_{h\times 1} = -\frac{\partial \mathcal{U}}{\partial \mathbf{y}}^{\mathrm{T}}(\Delta\mathbf{y}(t)) - \mathbf{D}\mathbf{p}(t) = \mathbf{p}(t) \tag{40}$$

These steady-state equations show that the error $\Delta\mathbf{y}(t) \rightarrow \mathbf{0}_{h\times 1}$ as $t \rightarrow \infty$ (Vidyasagar, 1993). ■

Remark 4. In our previous work (Navarro-Alarcon et al., 2013) we model the presence of measurement noise (common with vision sensors) as a disturbance rejection problem, and propose a simple method (Romero et al., 2013) to robustify a control action similar to (34).

Remark 5. The dynamic-state feedback action $\mathbf{p}(t)$ provides *smooth* trajectories to the robot manipulator. To see this property, consider the static-state feedback action $\boldsymbol{\phi}(t) = -\lambda\mathbf{D}^{-1}\Delta\mathbf{y}(t) \in \mathbb{R}^h$, for $\lambda > 0 \in \mathbb{R}$ and $\mathbf{D} \gg \lambda\mathbf{I}_{h\times h}$. By defining the potential function $\mathcal{U}(\Delta\mathbf{y}(t)) = \frac{1}{2}\lambda\Delta\mathbf{y}(t)\cdot\Delta\mathbf{y}(t)$, we can equivalently represent the variation $\frac{\mathrm{d}}{\mathrm{d}t}\mathbf{p}(t)$ as

$$\dot{\mathbf{p}}(t) = -\mathbf{D}(\mathbf{p}(t) - \boldsymbol{\phi}(t)) \tag{41}$$

which shows that $\mathbf{p}(t)$ (i.e. the solution of first-order system (41)) represents a smoothened version of $\boldsymbol{\phi}(t)$. This property is useful for vision-based velocity controllers since it helps to filter out noise from the image measurements.

4.4. Implementation algorithm

In Algorithm 1 we report the implementation of this control method. This algorithm presents a simple offline procedure to obtain a "good enough" initial vector of parameters $\widehat{\boldsymbol{\theta}}(0)$. This procedure consist in performing small and slow deformations around the initial configuration while running the online estimator. The purpose of doing this is to obtain a vector $\widehat{\boldsymbol{\theta}}(0)$ which provides the initial end-effector direction that minimizes $\Delta\mathbf{y}(t)$. This parameter initialization is useful in applications where large deviations from the ideal starting trajectory must be avoided.

The initialization of the state vector $\mathbf{p}(t)$ to a null vector allows to compute an incremental velocity command that can be tracked by the manipulator from a static configuration (this contrast with the traditional static-state feedback controller $\mathbf{v}(t) = -\lambda\widehat{\mathbf{J}}^{+}(t)\,\Delta\mathbf{y}(t)$ which provides the maximum velocity at $t = 0$). The estimator requires the measurement of the end-effector velocity, however, since the manipulator is kinematically controlled, we can alternatively use $\mathbf{v}(t-\delta t)$, for δt as the sampling rate. Note that low-pass filtering of the flow signals is needed to implement this method.

Algorithm 1 Flow-based deformation controller.

1: Move $\mathbf{v}(t) \leftarrow \mathbf{v}_{slow}$ to initialize $\widehat{\boldsymbol{\theta}}(0)$
2: Initialize $\mathbf{p}(0) \leftarrow \mathbf{0}_{h\times 1}$
3: **repeat**
4: Measure signals $\mathbf{x}(t)$, $\mathbf{s}(t)$, $\dot{\mathbf{x}}(t)$, $\dot{\mathbf{s}}(t)$
5: Low-pass filter $\dot{\mathbf{x}}(t)$, $\dot{\mathbf{s}}(t)$
6: Update parameters $\frac{\mathrm{d}}{\mathrm{d}t}\widehat{\boldsymbol{\theta}}(t) \leftarrow$ (27)
7: Compute state $\frac{\mathrm{d}}{\mathrm{d}t}\mathbf{p}(t) \leftarrow$ (36)
8: Compute controller $\mathbf{v}(t) \leftarrow$ (34)
9: Solve $\mathbf{v}(t)$ for $\boldsymbol{\omega}(t)$ and command motion
10: **until** Error $\|\Delta\mathbf{y}(t)\| < small\_val$

In this control implementation we present two task-specific values, $\mathbf{v}_{\mathrm{slow}} \in \mathbb{R}^3$ and $small\_val > 0 \in \mathbb{R}$. The vector $\mathbf{v}_{\mathrm{slow}}$ represents a "slow" velocity command that is large enough to slightly deform the object around the starting point $\mathbf{x}(0)$. The scalar *small_val* determines the minimum admissible deformation error.

5. Adaptive deformation controller

In this section we present an adaptive deformation controller that does not require the computation the deformation flow signals. This new approach differs from traditional visual servoing and kinematic controllers in that the control action is mapped to end-effector velocities by an *adaptively* varying transposed Jacobian-like matrix. In this method, the parameter's adaptation rule is not driven by a flow error, but by a combination of the explicit deformation error and an error of the estimated projection model. For clarity of presentation, we formulate this controller for a single point-based deformation, i.e. for a deformation feature vector defined as

$$\mathbf{y}(t) = \mathbf{s}_1(t) \in \mathbb{R}^2 \tag{42}$$

5.1. Projection estimation error

In contrast to the previous flow-based online estimator, we now define the *projection estimation error* (Liu et al., 2006) (Figure 4)

$$\begin{aligned}\boldsymbol{\epsilon}(t) &= \widehat{z}_1(t)\left(\widehat{\mathbf{s}}_1(t) - \mathbf{s}_1(t)\right) \in \mathbb{R}^2 \\ &= \left(\widehat{\mathbf{GC}}_1(t)\,\mathbf{x}(t) + \widehat{\mathbf{a}}_1(t)\right) \\ &\quad - \left(\widehat{\mathbf{g}_3^{\mathrm{T}}\mathbf{C}_1}(t)\,\mathbf{x}(t) + a_1\right)\mathbf{s}_1(t)\end{aligned} \tag{43}$$

Similar to the previous section, the terms with the upper symbol $\widehat{*}$ are constructed with the variable parameters $\widehat{\boldsymbol{\theta}}(t)$ and the scalars a_i.

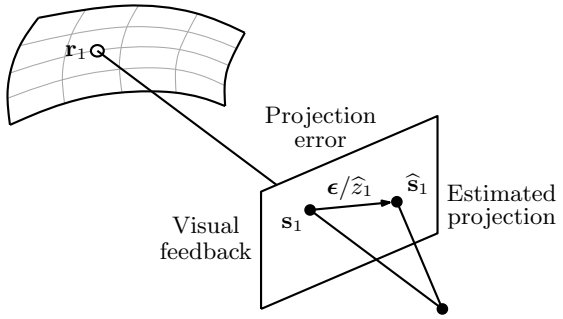


Fig. 4. Geometric representation of the projection estimation error $\boldsymbol{\epsilon}(t)$ (scaled by $1/\widehat{z}_1$).

As the top expression suggestively indicates, the error $\boldsymbol{\epsilon}(t)$ physically represents the scaled difference between the estimated perspective projection and the measured visual feedback.[2] We scale this error by the estimated depth $\widehat{z}_1(t)$ to obtain an expression that is linear with respect to the parameters' estimation error $\Delta\boldsymbol{\theta}(t)$. To see this, consider

$$\left(\mathbf{g}_3^{\mathrm{T}}\mathbf{C}_1\mathbf{x}(t) + a_1\right)\mathbf{s}_1(t) - \left(\mathbf{G}\mathbf{C}_1\mathbf{x}(t) + \mathbf{a}_1\right) = \mathbf{0}_{2\times 1} \quad (44)$$

which comes from multiplying the projection model (6) by the depth $z_1(t)$ and solving for $\mathbf{0}_{2\times 1}$. We add (44) to the error (43) to obtain

$$\begin{aligned}\boldsymbol{\epsilon}(t) &= \left(\left\{\widehat{\mathbf{G}\mathbf{C}}_1(t) - \mathbf{G}\mathbf{C}_1\right\}\mathbf{x}(t) + \left\{\widehat{\mathbf{a}}_1(t) - \mathbf{a}_1\right\}\right) \\ &\quad - \left(\left\{\widehat{\mathbf{g}_3^{\mathrm{T}}\mathbf{C}}_1(t) - \mathbf{g}_3^{\mathrm{T}}\mathbf{C}_1\right\}\mathbf{x}(t)\right)\mathbf{s}_1(t) \\ &= \mathbf{W}(\mathbf{x}(t), \mathbf{s}_1(t))\,\Delta\boldsymbol{\theta}(t) \end{aligned} \quad (45)$$

where $\mathbf{W}(\mathbf{x}(t), \mathbf{s}_1(t)) \in \mathbb{R}^{2\times 11}$ represents a known regression matrix, which does not dependent on $\boldsymbol{\theta}$ and has always rank 2. We compute this matrix as follows:

$$\mathbf{W}(\mathbf{x}(t), \mathbf{s}_1(t)) = \begin{bmatrix} \mathbf{x}^{\mathrm{T}}(t) & \mathbf{0}_{1\times 3} & -\mu_1(t)\,\mathbf{x}^{\mathrm{T}}(t) & 1 & 0 \\ \mathbf{0}_{1\times 3} & \mathbf{x}^{\mathrm{T}}(t) & -\nu_1(t)\,\mathbf{x}^{\mathrm{T}}(t) & 0 & 1 \end{bmatrix} \quad (46)$$

Assumption 5.1. Let $\mathbf{x}(t_j)$ and $\mathbf{s}_1(t_j)$ denote the vectors of end-effector position and visual feedback at the time instant t_j, respectively. Let us assume that during offline testing deformations (i.e. prior to the control experiments) we collect $P \geq 6$ non-collinear sets of these two vectors.

For each offline testing deformation, we compute its respective estimation errors

$$\begin{aligned}\boldsymbol{\epsilon}_j(t) &= \left(\widehat{\mathbf{G}\mathbf{C}}_1(t)\,\mathbf{x}(t_j) + \widehat{\mathbf{a}}_1(t)\right) \\ &\quad - \left(\widehat{\mathbf{g}_3^{\mathrm{T}}\mathbf{C}}_1(t)\,\mathbf{x}(t_j) + a_1(t)\right)\mathbf{s}_1(t_j) \end{aligned} \quad (47)$$

and construct the following extended error vector

$$\begin{aligned}\overline{\boldsymbol{\epsilon}}(t) &= \begin{bmatrix}\boldsymbol{\epsilon}^{\mathrm{T}}(t) & \boldsymbol{\epsilon}_1^{\mathrm{T}}(t) & \cdots & \boldsymbol{\epsilon}_P^{\mathrm{T}}(t)\end{bmatrix}^{\mathrm{T}} \\ &= \overline{\mathbf{W}}(\mathbf{x}(t), \mathbf{s}_1(t))\,\Delta\boldsymbol{\theta}(t) \in \mathbb{R}^{2+2P} \end{aligned} \quad (48)$$

where $\overline{\mathbf{W}}(\mathbf{x}(t), \mathbf{s}_1(t)) \in \mathbb{R}^{(2+2P)\times 11}$ represents the extended regression matrix which is defined by

$$\overline{\mathbf{W}}(\mathbf{x}(t), \mathbf{s}_1(t)) = \begin{bmatrix} \mathbf{W}(\mathbf{x}(t), \mathbf{s}_1(t)) \\ \mathbf{W}(\mathbf{x}(t_1), \mathbf{s}_1(t_1)) \\ \vdots \\ \mathbf{W}(\mathbf{x}(t_P), \mathbf{s}_1(t_P)) \end{bmatrix} \quad (49)$$

Note that once the P pairs of vectors $\mathbf{x}(t_j)$ and $\mathbf{s}_1(t_j)$ have been selected, the last P sub-matrices $\mathbf{W}(\mathbf{x}(t_j), \mathbf{s}_1(t_j))$ remain constant for all t, but not the adaptive parameters $\widehat{\boldsymbol{\theta}}(t)$ since they are continuously updated.

Remark 6. The extended regression matrix (49) has always rank 11 (Liu et al., 2006).

To better visualise this property, we re-arrange 11 constant row-vectors of $\overline{\mathbf{W}}(\mathbf{x}(t), \mathbf{s}_1(t))$ into the following 11×11 matrix

$$\begin{bmatrix} \mathbf{x}^{\mathrm{T}}(t_1) & \mathbf{0}_{1\times 3} & -\mu_1(t_1)\,\mathbf{x}^{\mathrm{T}}(t_1) & 1 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ \mathbf{x}^{\mathrm{T}}(t_6) & \mathbf{0}_{1\times 3} & -\mu_1(t_6)\,\mathbf{x}^{\mathrm{T}}(t_6) & 1 & 0 \\ \mathbf{0}_{1\times 3} & \mathbf{x}^{\mathrm{T}}(t_1) & -\nu_1(t_1)\,\mathbf{x}^{\mathrm{T}}(t_1) & 0 & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ \mathbf{0}_{1\times 3} & \mathbf{x}^{\mathrm{T}}(t_5) & -\nu_1(t_5)\,\mathbf{x}^{\mathrm{T}}(t_5) & 0 & 1 \end{bmatrix} \quad (50)$$

The linear independence of these 11 row-vectors is guaranteed by the fact that the P vectors of end-effector displacements $\mathbf{x}(t_j)$ and visual feedback $\mathbf{s}_1(t_j)$ are not collinear.

5.2. Transformed first-order plant

Instrumental for the new adaptive control design that we present in this section is to transform the differential expression (10) into a different representation. To this end, we first multiply (10) by $z_1(t)$ and then add the term $\frac{1}{2}\Delta\mathbf{s}_1(t)\,\dot{z}_1(t)$ to obtain

$$z_1(t)\,\dot{\mathbf{s}}_1(t) + \frac{1}{2}\Delta\mathbf{s}_1(t)\,\dot{z}_1(t) = \mathbf{A}(\mathbf{s}_1(t))\,\dot{\mathbf{x}}(t) \quad (51)$$

where the matrix $\mathbf{A}(\mathbf{s}_1(t)) \in \mathbb{R}^{2\times 3}$ is defined as

$$\mathbf{A}(\mathbf{s}_1(t)) = \mathbf{G}\mathbf{C}_1 - \frac{1}{2}\left(\mathbf{s}_1(t) + \mathbf{s}_1^*\right)\mathbf{g}_3^{\mathrm{T}}\mathbf{C}_1 \quad (52)$$

The rationale behind the definition of (51) is as follows: the scaling of the optical flow by the depth provides a system which is linear with respect to the unknown parameters $\boldsymbol{\theta}$; we can exploit this useful property to design adaptive controllers. The addition of the term $\frac{1}{2}\Delta\mathbf{s}_1(t)\,\dot{z}_1(t)$ allows us to design energy-motivated adaptive algorithms that do not require knowledge of the point's depth $z_1(t)$.

Remark 7. By definition, we know that the matrix (11) has always rank 3. This property means that the row vectors $\mathbf{g}_1^{\mathrm{T}}\mathbf{C}_1$, $\mathbf{g}_2^{\mathrm{T}}\mathbf{C}_1$, and $\mathbf{g}_3^{\mathrm{T}}\mathbf{C}_1$ are linearly independent. Therefore, after simple computations we can see that the "input" matrix $\mathbf{A}(\mathbf{s}_1(t))$ has always rank 2.

5.3. Adaptive kinematic controller

To implement the adaptive control method, we first construct with the vector of variable parameters $\widehat{\boldsymbol{\theta}}(t)$ an estimation of the unknown input matrix $\mathbf{A}(\mathbf{s}_1(t))$; we denote this adaptive matrix by $\widehat{\mathbf{A}}(t) \in \mathbb{R}^{2\times 3}$. To enforce a desired behavior to the transformed plant (51), we propose the velocity control input

$$\mathbf{v}(t) = -\widehat{\mathbf{A}}^{\mathrm{T}}(t)\, k_s \Delta\mathbf{s}_1(t) \tag{53}$$

where the vector $\Delta\mathbf{s}_1(t) = \mathbf{s}_1(t) - \mathbf{s}_1^* \in \mathbb{R}^2$ denotes the point deformation error, and $k_s > 0 \in \mathbb{R}$ represents a feedback scalar. Substitution of the control input (53) into the plant (51) yields

$$z_1(t)\,\dot{\mathbf{s}}_1(t) + \frac{1}{2}\Delta\mathbf{s}_1(t)\,\dot{z}_1(t) = -\mathbf{A}(\mathbf{s}_1(t))\,\widehat{\mathbf{A}}^{\mathrm{T}}(t)\, k_s \Delta\mathbf{s}_1(t) \tag{54}$$

To design the update rule for the parameters $\widehat{\boldsymbol{\theta}}(t)$, we rewrite (54) as follows:

$$z_1(t)\,\dot{\mathbf{s}}_1(t) + \frac{1}{2}\Delta\mathbf{s}_1(t)\,\dot{z}_1(t) = -\widehat{\mathbf{A}}(t)\,\widehat{\mathbf{A}}^{\mathrm{T}}(t)\, k_s \Delta\mathbf{s}_1(t) + \left(\widehat{\mathbf{A}}(t) - \mathbf{A}(\mathbf{s}_1(t))\right)\widehat{\mathbf{A}}^{\mathrm{T}}(t)\, k_s \Delta\mathbf{s}_1(t) \tag{55}$$

Note that the second term on the right-hand side of (55) can be equivalently expressed as a linear parameterization of the error $\Delta\boldsymbol{\theta}(t) = \widehat{\boldsymbol{\theta}}(t) - \boldsymbol{\theta}$

$$\left(\widehat{\mathbf{A}}(t) - \mathbf{A}(\mathbf{s}_1(t))\right)\widehat{\mathbf{A}}^{\mathrm{T}}(t)\, k_s \Delta\mathbf{s}_1(t) = \mathbf{Y}\left(\mathbf{s}_1(t), \widehat{\boldsymbol{\theta}}(t)\right)\Delta\boldsymbol{\theta}(t) \tag{56}$$

where $\mathbf{Y}(\mathbf{s}_1(t), \widehat{\boldsymbol{\theta}}(t)) \in \mathbb{R}^{2\times 11}$ represents a known regression matrix defined as

$$\mathbf{Y}\left(\mathbf{s}_1(t), \widehat{\boldsymbol{\theta}}(t)\right) = \begin{bmatrix} -\mathbf{v}^{\mathrm{T}}(t) & \mathbf{0}_{1\times 3} & \frac{1}{2}(\mu_1(t) + \mu_1^*)\,\mathbf{v}^{\mathrm{T}}(t) & 0 & 0 \\ \mathbf{0}_{1\times 3} & -\mathbf{v}^{\mathrm{T}}(t) & \frac{1}{2}(\nu_1(t) + \nu_1^*)\,\mathbf{v}^{\mathrm{T}}(t) & 0 & 0 \end{bmatrix} \tag{57}$$

with the vector $\mathbf{v}(t)$ defined as in (53).

To asymptotically minimize the point deformation error $\Delta\mathbf{s}_1(t)$, in this new control method we propose the parameter update rule

$$\dot{\widehat{\boldsymbol{\theta}}}(t) = -\boldsymbol{\Gamma}_2 \left(\mathbf{Y}^{\mathrm{T}}(t)\, k_s \Delta\mathbf{s}_1(t) + \overline{\mathbf{W}}^{\mathrm{T}}(t)\,\mathbf{K}_2\bar{\boldsymbol{\epsilon}}(t)\right) \tag{58}$$

where the diagonal and positive-definite matrices $\boldsymbol{\Gamma}_2 > 0 \in \mathbb{R}^{11\times 11}$ and $\mathbf{K}_2 > 0 \in \mathbb{R}^{(2+2P)\times(2+2P)}$ represent tuning matrices.

In the following proposition, we analyze the stability properties of the proposed adaptive deformation controller (53).

Proposition 5.2. *Consider that at the time instant t we count with $P \geq 6$ linearly independent vectors $\mathbf{x}(t_j)$ and $\mathbf{s}_1(t_j)$. For this situation, the control input* (53) *with parameter adaptation rule* (58) *enforces an energetically passive closed-loop system which asymptotically minimizes the point deformation error $\Delta\mathbf{s}_1(t)$.*

Proof. Considering the linear parameterizations (45) and (56), we can express the closed-loop system as

$$z_1(t)\,\dot{\mathbf{s}}_1(t) + \frac{1}{2}\Delta\mathbf{s}_1(t)\,\dot{z}_1(t) = -\widehat{\mathbf{A}}(t)\,\widehat{\mathbf{A}}^{\mathrm{T}}(t)\, k_s \Delta\mathbf{s}_1(t) + \mathbf{Y}(t)\,\Delta\boldsymbol{\theta}(t) \tag{59}$$

$$\dot{\widehat{\boldsymbol{\theta}}}(t) = -\boldsymbol{\Gamma}_2\overline{\mathbf{W}}^{\mathrm{T}}(t)\,\mathbf{K}_2\overline{\mathbf{W}}(t)\,\Delta\boldsymbol{\theta}(t) - \boldsymbol{\Gamma}_2\mathbf{Y}^{\mathrm{T}}(t)\, k_s \Delta\mathbf{s}_1(t) \tag{60}$$

To prove passivity of the closed-loop dynamical system given by (59)–(60), we introduce the energy-like scalar function

$$\mathcal{R}(t) = \frac{1}{2} z_1(t)\,\Delta\mathbf{s}_1^{\mathrm{T}}(t)\,\Delta\mathbf{s}_1(t) + \frac{1}{2}\Delta\boldsymbol{\theta}^{\mathrm{T}}(t)\,\boldsymbol{\Gamma}_2^{-1}\Delta\boldsymbol{\theta}(t) \in \mathbb{R} \tag{61}$$

whose time-derivative evaluated along trajectories of (59) and (60) satisfies

$$\begin{aligned} \dot{\mathcal{R}}(t) &= -k_s \Delta\mathbf{s}_1^{\mathrm{T}}(t)\,\widehat{\mathbf{A}}(t)\,\widehat{\mathbf{A}}^{\mathrm{T}}(t)\,\Delta\mathbf{s}_1(t) - \Delta\boldsymbol{\theta}^{\mathrm{T}}(t)\,\overline{\mathbf{W}}^{\mathrm{T}}(t)\,\mathbf{K}_2\overline{\mathbf{W}}(t)\,\Delta\boldsymbol{\theta}(t) \\ &\leq -c_w \|\Delta\boldsymbol{\theta}(t)\|^2 \\ &\leq 0 \end{aligned} \tag{62}$$

for $c_w > 0 \in \mathbb{R}$ as the minimum eigenvalue of the square matrix $\overline{\mathbf{W}}^{\mathrm{T}}(t)\,\mathbf{K}_2\overline{\mathbf{W}}(t) > 0$, which from Assumption 5.1 we know is positive definite. From the monotonic decrease of $\mathcal{R}(t)$, we prove that the closed-loop dynamical system is passive, moreover, dissipative with respect to the parameters' estimation error. This condition means that $\Delta\boldsymbol{\theta}(t) \to \mathbf{0}_{11\times 1}$ as $t \to \infty$. To prove asymptotic stability of the point deformation error, we analyze the equilibria of (59)–(60). These equations show that as $t \to \infty$, then $\Delta\mathbf{s}_1(t) \to \mathbf{0}_{2\times 1}$ (Vidyasagar, 1993). ■

Remark 8. For the highly over-constrained case where $P \gg 6$, the unknown parameters can be offline identified using standard least-squares fitting methods (see e.g. Higashimori et al., 2010). In our method, we use the testing deformation points $\mathbf{x}(t_j)$ and $\mathbf{s}_1(t_j)$ to continuously adapt the variable parameters, i.e. the estimated model is iteratively updated and not just fitted once, as with the former approach.

Remark 9. In contrast with the previous flow-based method, the adaptive deformation controller does not require to compute flow signals (via numerical differentiation) or to invert the deformation Jacobian matrix (in order to map the control action to end-effector velocities). However, note that its region of operation is restricted to the range where the affine model (3) approximates the point's deformation. The local nature of this approach does not

Algorithm 2 Adaptive deformation controller.

Collect P non-collinear vectors $\mathbf{x}(t_j), \mathbf{s}_1(t_j)$
Initialize $\widehat{\boldsymbol{\theta}}(0) \leftarrow \mathbf{i}$
repeat
  Measure signals $\mathbf{x}(t), \mathbf{s}_1(t)$
  Adapt parameters $\frac{\mathrm{d}}{\mathrm{d}t}\widehat{\boldsymbol{\theta}}(t) \leftarrow (58)$
  Compute controller $\mathbf{v}(t) \leftarrow (53)$
  Solve $\mathbf{v}(t)$ for $\boldsymbol{\omega}(t)$ and command motion
until Error $\|\Delta\mathbf{s}_1(t)\| < small\_val$

impose severe constraints to many real-world applications that require local or "small" deformations, e.g. the automatic positioning of elastic surfaces such as skin or textiles (Wada et al., 2001; Smolen and Patriciu, 2009; Torabi et al., 2009; Cusumano-Towner et al., 2011; Mallapragada et al., 2011).

5.4. Implementation algorithm

In Algorithm 2, we report the implementation of this adaptive controller. In contrast with the previous flow-based method, this controller requires no numerical differentiation and filtering of flow signals is needed. However, P testing points must be obtained before the control experiments. In this method, the parameter adaptation rule (58) can be initialized with an arbitrary non-zero vector $\mathbf{i} \in \mathbb{R}^l$. Note that a "rough" approximation of $\widehat{\boldsymbol{\theta}}(0)$ (which can be obtained from a previous experiment) or "small" arbitrary values (e.g. $\mathbf{i} = [0.1, \ldots, 0.1]^{\mathrm{T}}$) help to compute smooth initial trajectories to the manipulator. Similar to the previous algorithm, the scalar *small_val* determines the minimum admissible deformation error.

6. Experiments

6.1. Setup

The experimental study that we report in this section was conducted in the facilities of the Networked Sensors and Robotics Laboratory of the Chinese University of Hong Kong. The mechanical system that we use for the experiments is a TX-60 Stäubli robot manipulator, shown in Figure 5. This industrial manipulator has six revolute joints and a kinematic configuration similar to the well known PUMA robot (Lee and Ziegler, 1984).

To control the motion of the robot manipulator, we use an open architecture motion controller (Pertin and Bonnet des Tuves, 2004) that allow us to explicitly set the angular velocity of each joint in real-time (RT). We use a RT-Linux computer (with a Xenomai-patched kernel, see Gerum (2004)) to process the sensor's feedback signals and to compute the joint velocity command $\boldsymbol{\omega}(t)$. Then, we transmit the angular velocity vector via TCP/IP communication to the manipulator's low-level servo controller. All control algorithms are implemented with a deterministic (i.e. with a strict constant behavior) servo-loop of $\delta t = 4$ ms.

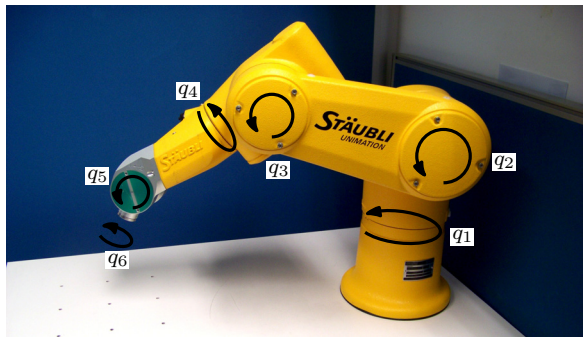


Fig. 5. The TX-60 Stäubli robot manipulator used for our experimental study.

We conduct this study with different types of soft objects and deformation features. To construct the deformation feature vector, we place artificial markers on the objects' surface (see Figures 6 and 15, below). In our formulation, the uncertain location of these feature points corresponds to $\mathbf{r}_i(t)$. To visually track the position of these markers, we use the Lucas–Kanade algorithm (Baker and Matthews, 2004) that comes implemented with the OpenCV libraries (Bradski, 2000). At the beginning of each experiment, we manually select the k feedback visual points $\mathbf{s}_i(t)$. To acquire the visual feedback of the scene, we use a Logitech C210 camera.

6.2. General deformation features

We test the performance of the flow-based control method presented in Section 4 with the two deformation tasks shown in Figure 6. For the task in Figure 6(a), we locally quantify the deformation of the compliant beam-like object (Liu and Sun, 2000) through the following 2-dimensional ($h = 2$) deformation feature vector

$$\mathbf{y}(t) = \begin{bmatrix} \frac{1}{2}\frac{\|\mathbf{s}_1(t)-\mathbf{s}_3(t)\|}{\|\mathbf{s}_j(t)-\boldsymbol{\pi}(t)\|} & \operatorname{atan}\left(\frac{\mu_2(t)-\mu_2(t)}{\nu_2(t)-\nu_1(t)}\right) \end{bmatrix}^{\mathrm{T}} \tag{63}$$

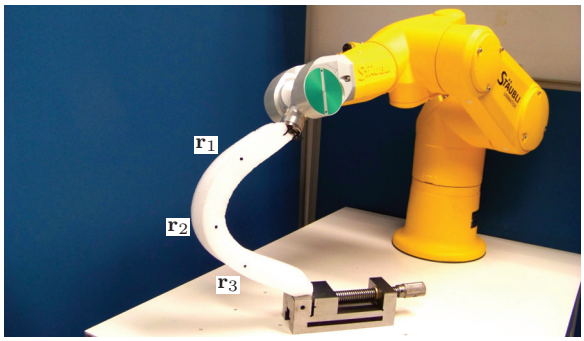
The first coordinate of $\mathbf{y}(t)$ represents a normalized (by the radius) quasi-curvature metric of the circumference that passes through the points $\mathbf{s}_1(t)$, $\mathbf{s}_2(t)$, and $\mathbf{s}_3(t)$, and with the origin at $\boldsymbol{\pi}(t) \in \mathbb{R}^2$. Using standard geometric methods, we compute the *circumcenter* $\boldsymbol{\pi}(t)$ as follows (Kimberling, 1998):

$$\boldsymbol{\pi}(t) = \frac{1}{2B(t)} \begin{bmatrix} \nu_2-\nu_3 & \nu_3-\nu_1 & \nu_1-\nu_2 \\ \mu_3-\mu_2 & \mu_1-\mu_3 & \mu_2-\mu_1 \end{bmatrix}(t) \times \begin{bmatrix} \|\mathbf{s}_1(t)\|^2 & \|\mathbf{s}_2(t)\|^2 & \|\mathbf{s}_3(t)\|^2 \end{bmatrix}^{\mathrm{T}} \tag{64}$$

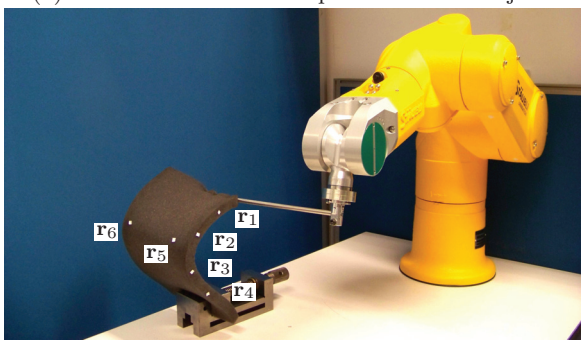
for a scalar function $B(t) \in \mathbb{R}$ defined as

$$B(t) = \mu_1(t)\big(\nu_2(t)-\nu_3(t)\big) + \mu_2(t)\big(\nu_3(t)-\nu_1(t)\big) + \mu_3(t)\big(\nu_1(t)-\nu_2(t)\big) \tag{65}$$

For this unitless metric, a value close to zero describes flatness, a value close to one approximates a circle with



(a) Interaction with a compliant "linear" object.



(b) Interaction with a soft foam sheet.

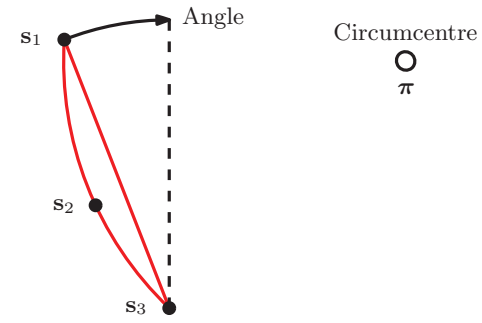
Fig. 6. The robot manipulator interacting with testing deformable objects. Note the artificial markets that we place on the object's surface to track its position with a vision system. (a) The linear object (with dimensions of $4 \times 5 \times 40$ mm) is directly attached to the end-effector. (b) The foam sheet (with dimensions of $2 \times 18 \times 30$ mm) is manipulated with a metal rod connected to the robot.

radius $\frac{1}{2}\|\mathbf{s}_j(t) - \boldsymbol{\pi}(t)\|$. The second coordinate of (63) represents the angle of the line joining the points $\mathbf{s}_1(t)$ and $\mathbf{s}_3(t)$ with respect to the vertical coordinates. Physically, the features (63) approximate the linear object's "bending" and its relative orientation. See Figure 7(a) for a conceptual representation of these feedback deformation features.

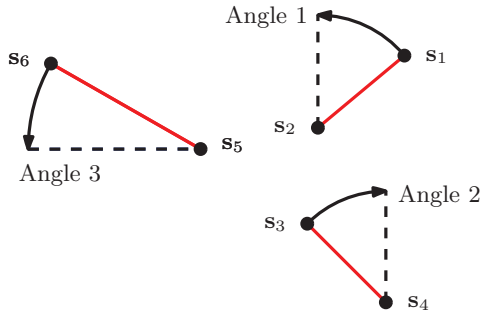
For the task shown in Figure 6(b), we quantify the deformation of a soft foam sheet with a 3-dimensional feature vector constructed with $k = 6$ visual feedback points. We locally define this deformation vector by

$$\mathbf{y}(t) = \begin{bmatrix} \operatorname{atan}\left(\frac{\mu_2(t)-\mu_2(t)}{\nu_2(t)-\nu_1(t)}\right) \\ \operatorname{atan}\left(\frac{\mu_4(t)-\mu_3(t)}{\nu_4(t)-\nu_3(t)}\right) \\ \operatorname{atan}\left(\frac{\nu_6(t)-\nu_5(t)}{\mu_6(t)-\mu_5(t)}\right) \end{bmatrix} \tag{66}$$

The coordinates of (66) represent angles of lines computed with the six visual feedback points $\mathbf{s}_i(t)$. Physically, these features approximate the object's 3-dimensional bending and orientation. Figure 7(b) conceptually depicts these features.



(a) Curvature and angle features.



(b) Three angle features.

Fig. 7. Conceptual representation of the deformation features for the setup shown in Figure 6.

6.3. *Experiments with the flow-based deformation controller*

In our system, the vectors $\mathbf{x}(t)$ and $\mathbf{s}(t)$ are measured in units of meters and in pixels, respectively. We use the following tuning matrix $\boldsymbol{\Gamma}_1$ to compensate for different unit and motion scales:

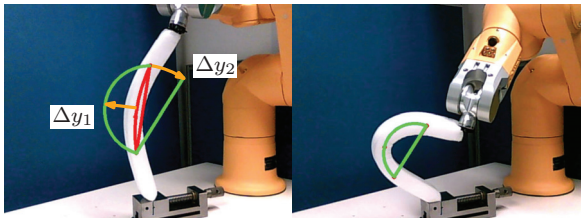
$$\boldsymbol{\Gamma}_1 = \begin{bmatrix} \mathbf{N}_1 & \cdots & \mathbf{0}_{11\times 11} \\ \vdots & \ddots & \vdots \\ \mathbf{0}_{11\times 11} & \cdots & \mathbf{N}_k \end{bmatrix} \tag{67}$$

where the diagonal submatrix $\mathbf{N}_j \in \mathbb{R}^{11\times 11}$ is defined for each point $\mathbf{s}_j(t)$ as

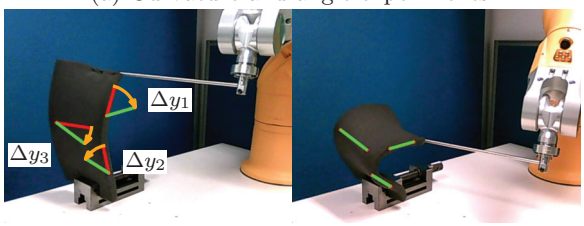
$$\mathbf{N}_j = \begin{bmatrix} 5000\mathbf{I}_{6\times 6} & \mathbf{0}_{6\times 5} \\ \mathbf{0}_{5\times 6} & 0.1\mathbf{I}_{5\times 5} \end{bmatrix}, \qquad \text{for } j = 1, \cdots, k \tag{68}$$

The use of the "small" matrix $0.1\mathbf{I}_{5\times 5}$ also helps to make the online estimator less sensitive to the noisy terms that come from the derivative of the image measurements (see the definition of the regression matrix $\mathbf{F}_i(t)$ in (26)). Additionally, note that this small matrix directly regulates the responsiveness (i.e. how fast it varies) of the scalar $\widehat{z}_i(t)$; generally, a "slow" variation of this scalar is preferred as it helps to avoid zero crossings (see the computation of the Jacobian matrix in (32)).

For the two tasks depicted in Figure 6, we use the matrix $\mathbf{K}_1 = \mathbf{I}_{2k\times 2k}$. Different values of this matrix, e.g. $\mathbf{K}_1 = \operatorname{diag}(\mathbf{I}_{2\times 2}, 0.1\mathbf{I}_{2\times 2}, \ldots, 0.5\mathbf{I}_{2\times 2})$, can be used to individually tune the flow estimation of each point $\mathbf{s}_i(t)$. Before conducting the control experiments, we initialize the variable



(a) Curvature and angle experiments.



(b) Angle deformation experiments.

Fig. 8. Snapshots of the initial and final configuration of the deformable objects obtained with the flow-based deformation controller. (a) Experiments with two deformation DOF. The left figure depicts an initial curvature and angle deformation errors of $\Delta y_1(0) = 0.7$ and $\Delta y_2(0) = 0.4$ rad, respectively. (b) Experiments with three deformation DOF. The left figure depicts initial angle errors of $\Delta y_1(0) = 1$ rad, $\Delta y_2(0) = 0.8$ rad, and $\Delta y_3(0) = 0.5$ rad.

parameters $\widehat{\boldsymbol{\theta}}(t)$ with a vector of 1s, and then follow the initialization procedure described in Algorithm 1.

In our study, we implement the following potential control action

$$\frac{\partial \mathcal{U}}{\partial \mathbf{y}}^{\mathrm{T}}(\Delta\mathbf{y}(t)) = K_y \mathbf{sat}(\Delta\mathbf{y}(t)) \tag{69}$$

with a feedback gain $K_y = 3$, and a saturation function $\mathbf{sat}(\cdot) \in \mathbb{R}^h$ that satisfies $-\boldsymbol{\alpha} \leq \mathbf{sat}(\cdot) \leq \boldsymbol{\alpha}$ for $\boldsymbol{\alpha} \in \mathbb{R}^h$ as a constant bound vector. We use the following bounding scalars: $\alpha_i = 0.25$ and $\alpha_j = 0.15$ rad for the unitless curvature and angle features, respectively. We compute the dynamic-state velocity control action $\mathbf{p}(t)$ with a dissipation matrix of $\mathbf{D} = 10\mathbf{I}_{h\times h}$.

To filter out noise from the measured flow signals, here generically denoted by $\mathbf{f}_m(t)$, we make use of the following first-order filter:

$$\frac{\mathrm{d}}{\mathrm{d}t}\mathbf{f}(t) = -\gamma(\mathbf{f}(t) - \mathbf{f}_m(t)) \tag{70}$$

where $\gamma > 0 \in \mathbb{R}$ represents the smoothing factor, and $\mathbf{f}(t)$ denotes the filtered signal.

We now report the results obtained with this control method. Figure 8 shows snapshots of the conducted experiments. These figures show the initial (undeformed) and final (deformed) configurations of both compliant objects. The overlaid red curves represent the deformation features, which we compute with the real-time visual feedback points $\mathbf{s}_i(t)$. The overlaid green curves represent the reference of each deformation coordinate $\mathbf{y}(t) = [y_1(t), \ldots, y_h(t)]^{\mathrm{T}}$.

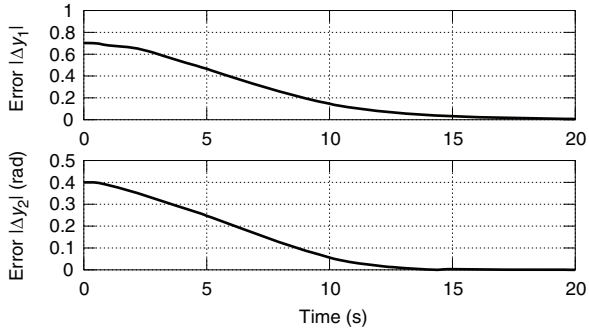


Fig. 9. Magnitude of the deformation error coordinates $\Delta\mathbf{y}(t) = [\Delta y_1(t), \Delta y_2(t)]^{\mathrm{T}}$ of the curvature-angle deformation task shown in Figure 6(a). The first coordinate of this vector represents the unitless curvature metric whereas the second represents an angle, as defined in (63).

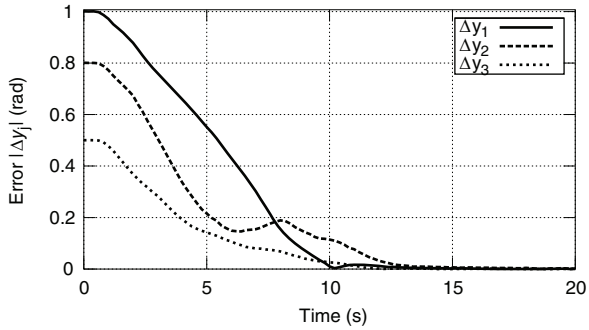


Fig. 10. Magnitude of the three deformation error coordinates $\Delta\mathbf{y}(t) = [\Delta y_1(t), \Delta y_2(t), \Delta y_3(t)]^{\mathrm{T}}$ of three-angle deformation task shown in Figure 6(b) and defined in (66).

Figures 9 and Figure 10 show the magnitude of the deformation errors $\Delta y_j(t)$. For these two and three DOF experiments, in Figure 11 and Figure 12 we present a qualitative comparison of the visually measured deformation flow vector $\dot{\mathbf{y}}(t)$ and the flow vector $\mathbf{m}(t) \in \mathbb{R}^h$ computed (in real-time) by

$$\mathbf{m}(t) = \widehat{\mathbf{J}}(t)\,\dot{\mathbf{x}}(t) \tag{71}$$

From these figures we see that the estimated matrix $\widehat{\mathbf{J}}(t)$ closely reconstructs the unknown deformation mapping $\dot{\mathbf{x}}(t) \mapsto \dot{\mathbf{y}}(t)$. This graphical comparison corroborates that our online estimation method is suitable for kinematic control of manipulators. We can also see that the method removes noise from the measurement signal $\mathbf{s}(t)$ and the numerical differentiation of the optical flow $\dot{\mathbf{s}}(t)$. Finally, Figure 13 and Figure 14 depict the trajectory of the end-effector position $\mathbf{x}(t) = [x_1(t), x_2(t), x_3(t)]^{\mathrm{T}}$ during the deformation experiments.

6.4. *Experiments with the adaptive deformation controller*

To test the performance of the adaptive control method presented in Section 5, we use the experimental setup shown in

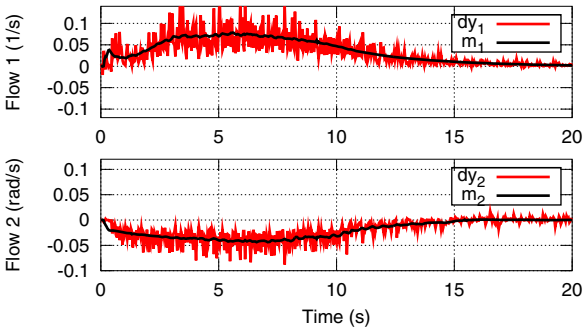


Fig. 11. Comparison of the visually measured deformation flow $\dot{\mathbf{y}}(t)$ (red line) and the flow computed by $\mathbf{m}(t) = \widehat{\mathbf{J}}(t)\,\dot{\mathbf{x}}(t)$ (black line), for the curvature-angle deformation task shown in Figure 6(a).

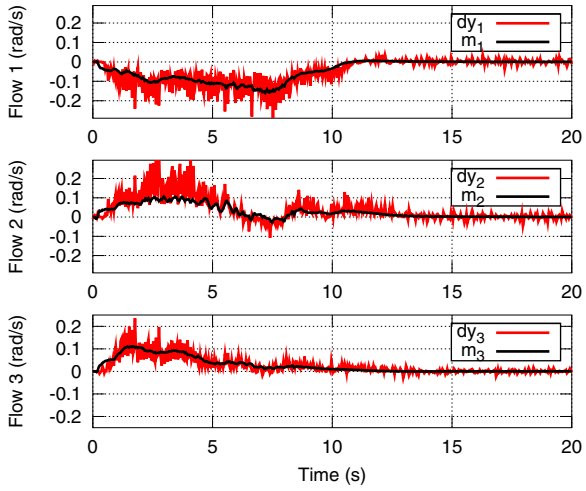


Fig. 12. Comparison of the visually measured deformation flow $\dot{\mathbf{y}}(t)$ (red line) and the flow computed by $\mathbf{m}(t) = \widehat{\mathbf{J}}(t)\,\dot{\mathbf{x}}(t)$ (black line), for the three-angle deformation task shown in Figure 6(b).

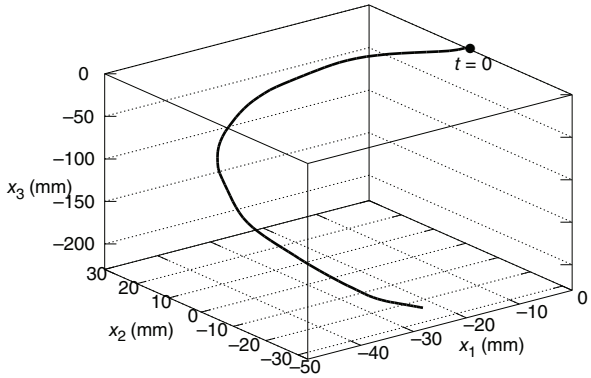


Fig. 13. Spatial displacement of the end-effector Cartesian coordinates $x_j(t)$ during the curvature-angle deformation task shown in Figure 6(a).

Figure 15. With this control approach we only regulate the displacements of a single visual feedback point,[3] in other words we define $\mathbf{y}(t) = \mathbf{s}_1(t) \in \mathbb{R}^2$. In our experimental study, we first perform offline testing deformations with the

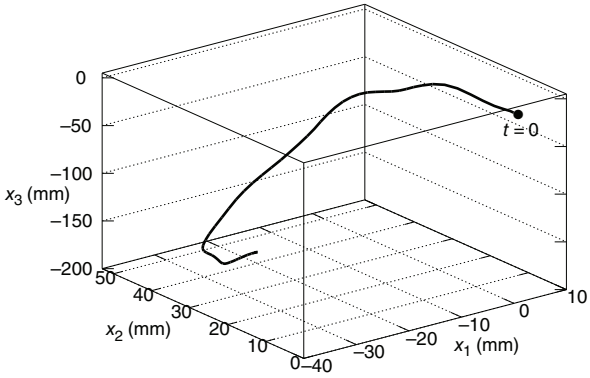


Fig. 14. Spatial displacement of the end-effector Cartesian coordinates $x_j(t)$ during the three-angle deformation task shown in Figure 6(b).

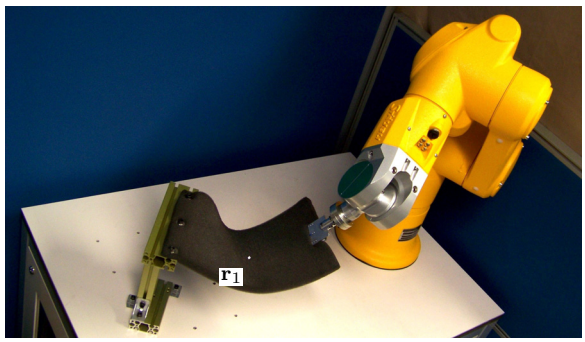


Fig. 15. The setup for the adaptive control experiments. We use the same object as in Figure 8(a). In this experimental study, we place only one point on the surface of the object.

elastic object. To this end, we command the manipulator to deform the object into different and representative configurations around the desired operating point. This way, we collect $P = 6$ image points $\mathbf{s}_1(t_j)$ and their respective position vectors $\mathbf{x}(t_j)$. Figure 16 shows the location in the image plane of the P offline points, as well as the start and target image positions.

To compute the adaptive control law (53), we use a proportional feedback gain of $k_s = 0.00003$. Note that since this method does not invert the Jacobian matrix, the proportional gain k_s must be much smaller than the gain K_y of the control action (69). In the experiments, we compute the parameter adaptation rule (58) with a tuning matrix $\mathbf{K}_2 = 300\mathbf{I}_{14\times 14}$; note that the use of different values, e.g. $\mathbf{K}_2 = \text{diag}(\,100\mathbf{I}_{2\times 2}, \mathbf{I}_{2\times 2}, \ldots, \mathbf{I}_{2\times 2})$, gives different priorities to specific perspective projection errors. This feature can be used, for example, to make the adaptive algorithm more sensitive with respect to the current error (43).

In this study, we implement the parameter tuning matrix

$$\boldsymbol{\Gamma}_2 = \begin{bmatrix} 0.5\mathbf{I}_{6\times 6} & \mathbf{0}_{6\times 5} \\ \mathbf{0}_{5\times 6} & 0.0001\mathbf{I}_{5\times 5} \end{bmatrix} \tag{72}$$

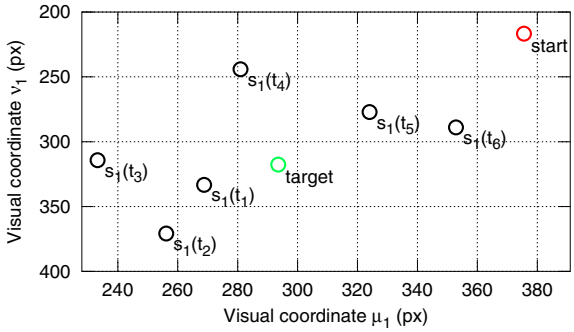


Fig. 16. Image location of the start and target configurations, and the $P = 6$ offline points $\mathbf{s}_1(t_j)$ that are obtained with the displacements $\mathbf{x}(t_1) = [0.562, 0.187, -0.191]$, $\mathbf{x}(t_2) = [0.504, 0.073, -0.157]$, $\mathbf{x}(t_3) = [0.438, 0.087, -0.103]$, $\mathbf{x}(t_4) = [0.463, 0.166, -0.147]$, $\mathbf{x}(t_5) = [0.513, 0.238, -0.135]$, and $\mathbf{x}(t_6) = [0.458, 0.180, -0.134]$, all given in meters.

whose values are chosen in order to compensate for the different unit scales between meters and pixels (see the definition of the regression matrix (46)).

Figure 17 shows snapshots of the initial and final configurations of this indirect point positioning experiment. In this figure, the overlaid red circle represents the real-time visual feedback of the deformable point, whereas the overlaid green circle represents the constant target point.

The results of this deformation experiment are as follows: Figure 18 shows the magnitude of both deformation error coordinates $\Delta y_j(t)$. From this figure we can see that the profiles of the deformation errors present an exponential decay, which contrast with the saturated response of the control action (69). Figure 19 depicts the trajectory of end-effector Cartesian displacements obtained during the experiment. From this figure we can see that the operating range of the point deformation task can be fairly approximated by the linear deformation model (3) (compare the almost straight curve in Figure 19 with those in Figure 13 and Figure 14).

6.5. Comparison of control laws

In this study we compare the performance of three different control approaches, namely the adaptive controller (53), the proposed dynamic-state feedback control law (34) (i.e. flow-based method), and its static-state feedback variant

$$\mathbf{v}(t) = -\lambda \widehat{\mathbf{J}}^{+}(t)\, \Delta\mathbf{y}(t) \tag{73}$$

which closely resembles the classical image-based kinematic control, as presented in Chaumette and Hutchinson (2006). Note that in the control input (73) we also compute the deformation Jacobian matrix with the flow-based rule (32).

To qualitatively compare the performance of these control methods, we use the same point deformation task shown in Figure 17. We implement the kinematic controllers (34) and (73) with the following gains: $\mathbf{D} = 5\mathbf{I}_{2\times 2}$, $K_y = 3$, and $\lambda = 0.4$ (these gains were selected to provide a

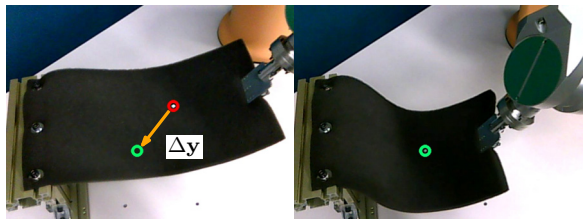


Fig. 17. Snapshots of the object's initial and final configurations obtained with the adaptive deformation controller.

"similar" response as the one obtained with the adaptive controller in Figure 18). The proportional action (69) is implemented with the bounding vector $\boldsymbol{\alpha} = [20, 20]^{\mathrm{T}}$. The online Jacobian estimator is implemented with tuning gains as given in (68).

We now present the results of this comparative study. Figure 20 shows the time-evolution of the norm $\|\Delta\mathbf{y}(t)\|_2 = \sqrt{\Delta y_1^2(t) + \Delta y_2^2(t)}$ obtained with the deformation controllers. In this figure, the curve A is obtained with the adaptive control method (53); the curve D is obtained with the dynamic-state feedback control law (34); the curve S is obtained with the static-state feedback control law (73). For these experiments, Figure 21 depicts the resulting visual error trajectories (i.e. curves in the image plane) obtained with the three control approaches. Note that in this comparative study, we set the gains such that the error $\|\Delta\mathbf{y}(t)\|_2$ converges in a similar manner (at around 12 s).

We now qualitatively compare the profiles of the velocity control input $\mathbf{v}(t)$ computed with these deformation controllers. Figure 22 shows the three components of the Cartesian velocity input vector $\mathbf{v}(t) = [v_1(t), v_2(t), v_3(t)]^{\mathrm{T}}$ computed with the adaptive controller. Similarly, Figure 23 and Figure 24 depict the velocity input $\mathbf{v}(t)$ computed with the dynamic-/static-state feedback control laws, respectively. These figures show that the two kinematic controllers proposed in this paper compute incremental (starting at/close to $\mathbf{0}_{3\times 1}$) and smooth velocity inputs to the manipulator (compare these profiles with Figure 24). This feature is particularly useful for kinematically controlled mechanical systems since inaccurate visual tracking algorithms may generate discontinuous velocity commands. From these figures we can also see that the velocity input computed with the adaptive method presents less noise compared with the flow-based method (for the latter method, the major source of noise comes from the time-differentiation of the image measurements).

6.6. Comparison of online Jacobian estimators

In this section we compare three different algorithms that compute the deformation Jacobian matrix in real-time. We compare the accuracy of the online estimator proposed in this paper (which considers the perspective camera model), with the estimator reported in Navarro-Alarcon et al. (2013)

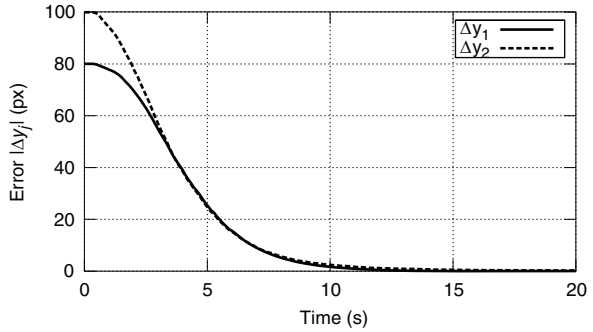


Fig. 18. Magnitude of the error coordinates $\Delta\mathbf{y}(t) = [\Delta y_1(t), \Delta y_2(t)]^{\mathrm{T}}$ of point deformation task shown in Figure 17.

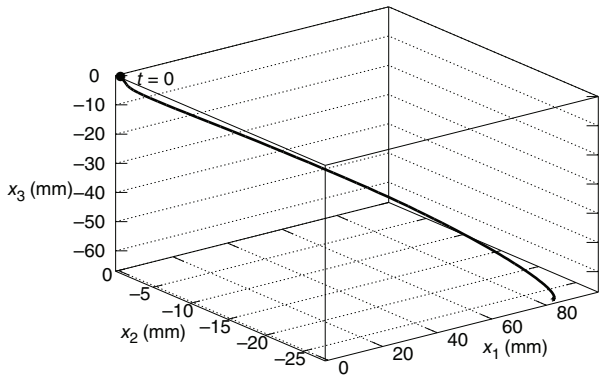


Fig. 19. Spatial displacement of the end-effector Cartesian coordinates $x_j(t)$ during the point deformation task shown in Figure 17.

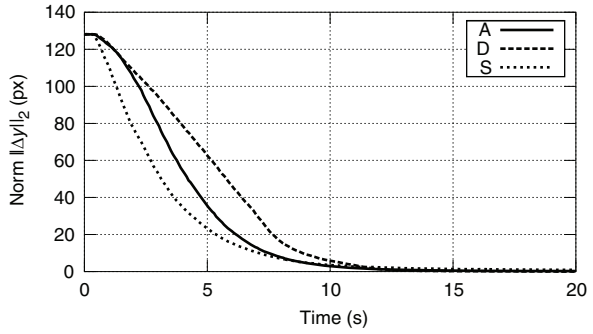


Fig. 20. Time-evolution of the norm $\|\Delta\mathbf{y}(t)\|_2$ obtained with the adaptive (curve A), dynamic-state feedback (curve D), and static-state feedback (curve S) deformation controllers.

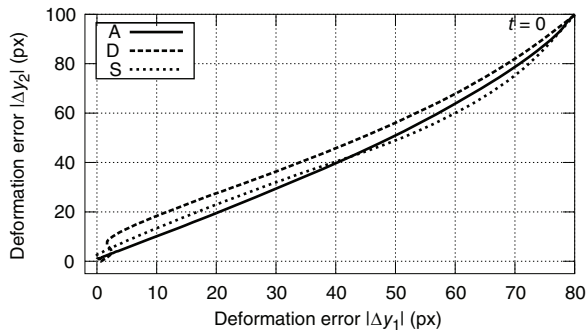


Fig. 21. Comparison of the deformation error trajectories obtained with the adaptive controller (curve A), the dynamic-state feedback control law (curve D), and the static-state feedback control law (curve S).

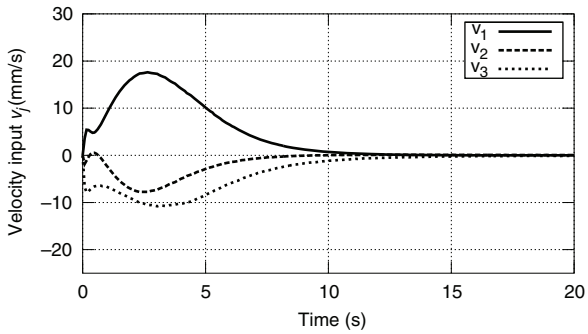


Fig. 22. Components of the velocity control input $\mathbf{v}(t) = [v_1(t), v_2(t), v_3(t)]^{\mathrm{T}}$ computed with the adaptive control method.

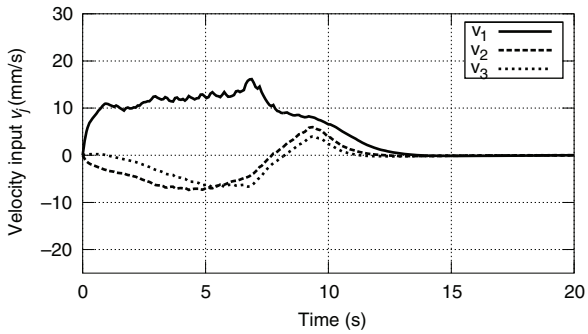


Fig. 23. Components of the velocity control input $\mathbf{v}(t)$ computed with the control law (34).

(which is based on the Broyden update rule), and the estimator reported in Navarro-Alarcon and Liu (2013) (which considers the affine camera model).

The aim of this study is to qualitatively compare the output deformation flow $\dot{\mathbf{y}}(t)$ computed with these three methods. For that, the manipulator's end-effector deforms the soft object in a feed-forward manner (i.e. no closed-loop deformation control is performed) while the three algorithms (namely the perspective, Broyden, and affine) simultaneously reconstruct the deformation flow. We conduct this comparative study with the same point deformation setup shown in Figure 15.

To provide a "similar" response to the one obtained with the perspective estimator (27) (whose gains are given in (68)), we implement the Broyden update rule with a learning gain $\Gamma_B = 0.02$; for details about the method see Navarro-Alarcon et al. (2013). Similarly, we tune the affine estimator with a matrix $\mathbf{\Gamma}_A = 10000\mathbf{I}_{l\times l}$; this tuning matrix has the same purpose as $\mathbf{\Gamma}_2$ in (27), see Navarro-Alarcon and Liu (2013) for details about its implementation.

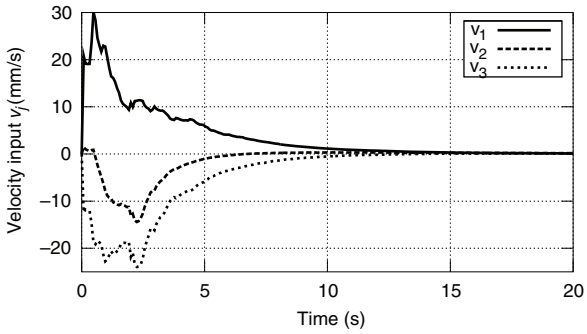


Fig. 24. Components of the velocity control input $\mathbf{v}(t)$ computed with the control law (73).

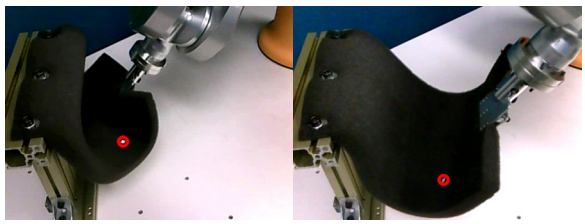


Fig. 25. Snapshots of the trajectory followed by the manipulator during the Jacobian estimation experiments.

Figure 25 depicts snapshots of the point deformation task performed by the robot manipulator. The Cartesian trajectory followed by the manipulator's end-effector is shown in Figure 26. Figure 27 shows a comparison of the visually measured deformation flow $\dot{\mathbf{y}}(t)$ and the flow $\mathbf{m}(t) = \widehat{\mathbf{J}}(t)\,\dot{\mathbf{x}}(t)$ computed with the online estimator for perspective cameras that we present in this paper. Similarly, Figure 28 and Figure 29 show the results obtained with the Broyden update rule and the affine estimator, respectively. From this simple graphical comparison we can see that the method in Navarro-Alarcon and Liu (2013) is relatively less susceptible to noise than the online estimator (27) and the Broyden update rule (compare the profiles amongst the different flow signals $\mathbf{m}(t)$, i.e. the black lines). Certainly, tuning gains play a major role in determining the algorithm's response; however, in our experiments we noted that the much simpler affine method provided a smoother approximation of the Jacobian matrix.

6.7. Multimedia material

Videos of the uncalibrated experiments shown in Figure 8 and 17 are included in Multimedia Extension 1.

7. Conclusion and future work

In this paper, we have proposed two new kinematic controllers for robot manipulators performing active deformation tasks with unknown elastic objects. We first presented a control method that estimates the deformation Jacobian matrix in real-time; this method minimizes the error between measured and estimated deformation flows.

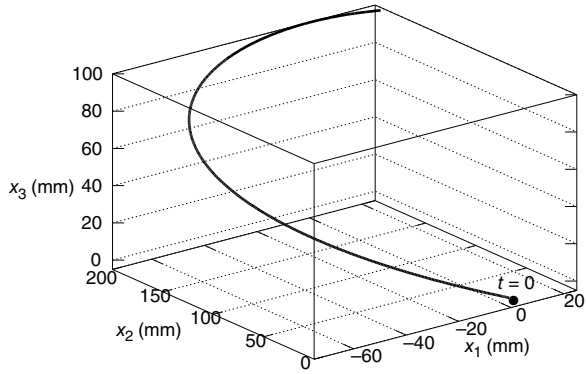


Fig. 26. Spatial displacement of the end-effector Cartesian coordinates $x_j(t)$ during the Jacobian estimation experiments.

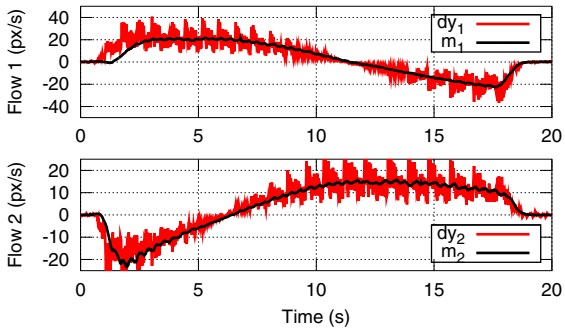


Fig. 27. Comparison of the visually measured deformation flow $\dot{\mathbf{y}}(t)$ (red line) and the flow computed by $\mathbf{m}(t) = \widehat{\mathbf{J}}(t)\,\dot{\mathbf{x}}(t)$ (black line) with the perspective estimator (27).

Next we presented a controller that combines offline information with online adaptation of variable parameters; this method does not require to numerically compute flow signals. Finally, we reported experimental results to evaluate the performance of our new controllers.

There are some important considerations that must be taken into account when implementing deformation controllers. For example, a critical issue is that the desired deformation task must be physically realizable. This condition simply means that the target configuration must be achievable with the end-effector "gripper", the object's geometry, and the camera's setup being used. If this condition is not met, the Jacobian matrix will most likely lose rank (consequently, it cannot be inverted, and its transpose maps to null velocity).

There are also a number of limitations for the controllers that we proposed. For example, the use of the optical flow in our first method can be a disadvantage since numerical differentiation techniques are usually noisy. To cope with this issue, in this paper we proposed to low-pass filter the computed flow signals. However, note that the filtering process may introduce some delay, hence affecting the performance; for this method slow motion is recommended.

As for the second control method, the local nature of the affine deformation model can also be a practical limitation

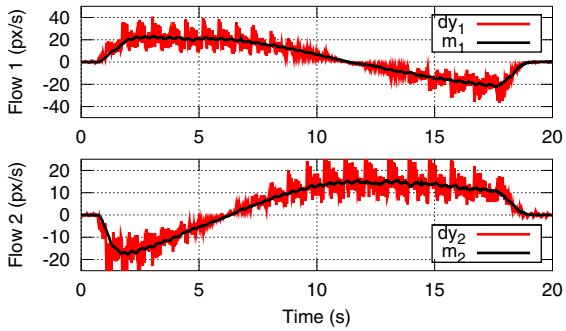


Fig. 28. Comparison of the visually measured deformation flow $\dot{\mathbf{y}}(t)$ (red line) and the flow computed by $\mathbf{m}(t) = \widehat{\mathbf{J}}(t)\,\dot{\mathbf{x}}(t)$ (black line) with the Broyden update rule as in Navarro-Alarcon et al. (2013).

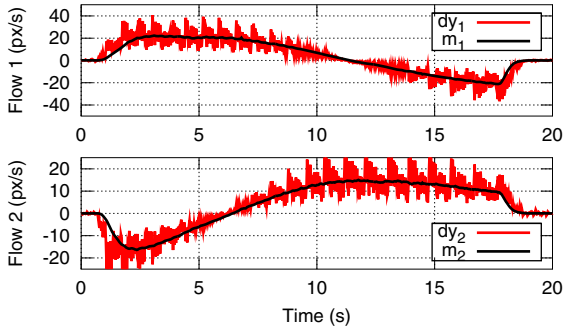


Fig. 29. Comparison of the visually measured deformation flow $\dot{\mathbf{y}}(t)$ (red line) and the flow computed by $\mathbf{m}(t) = \widehat{\mathbf{J}}(t)\,\dot{\mathbf{x}}(t)$ (black line) with the affine estimator of Navarro-Alarcon and Liu (2013).

for tasks that require a wide range of nonlinear deformations; for this method "small" or local deformations are recommended. The major drawback of the adaptive controller with respect to the flow-based method is the need to obtain offline information about the object's deformation. However, since the adaptive method does not require to numerically compute the flow signals, its performance is less sensitive to image noise. In our experimental study, the adaptive controller provided smoother and more stable deformation trajectories.

In the presented comparative study for the online Jacobian estimators, we could see that our new algorithm seems to be more affected by noise than our previous method for affine cameras. This limitation arises with the explicit use of the optical flow within the regression matrix $\mathbf{Q}(t)$ (something that is not needed with our earlier method). In the conducted experiments, the estimator for affine cameras provided better approximations of the Jacobian matrix than with our new method. Therefore, further research is needed in order to improve the performance of the proposed online algorithm.

To the best of our knowledge, the analytical methods presented in Sections 4 and 5 are both original. We think that the ideas behind these methods can be used in other types of applications. For example, a "similar" approach as with the flow-based algorithm can be used to compute the Jacobian matrix for visual servoing tasks in eye-in-hand configurations. The singularity-free adaptive controller can be used in classical (i.e. with no deformation involved) fixed-camera visual servoing applications.

The use of velocity observers seems an attractive future direction to improve the performance of the proposed flow-based method. The partition of a deformation trajectory into several smaller steps or regions seems a feasible solution to local nature of the adaptive controller. Another interesting future direction is to consider that not only linear displacements but also rotations of the end-effector produce deformations on the object. By doing this we can increase the dimension of the Jacobian matrix kernel and thus avoid singular configurations. Finally, future research is needed to incorporate nonlinear parametric deformation models, e.g. as proposed in Berenson (2013), within our current formulation.

Funding

This work was supported by the Hong Kong Research Grants Council (RGC) (grant numbers 415011 and CUHK6/CRF/13G); the Hong Kong Innovation and Technology Fund (ITF) (grant number ITS/475/09) and the Shun Hing Institute of Advanced Engineering.

Notes

1. This property also applies to the camera's projection matrix $\mathbf{M}$.
2. In this expression, we define $\widehat{z}_1(t) = \widehat{\mathbf{g}_3^{\mathrm{T}}\mathbf{C}_1}(t)\,\mathbf{x}(t) + a_1$ and $\widehat{\mathbf{s}}_1(t) = \frac{1}{\widehat{z}_1(t)}(\widehat{\mathbf{GC}}_1(t)\,\mathbf{x}(t) + \widehat{\mathbf{a}}_1(t))$.
3. This deformation task is similar to the one reported in Hirai and Wada (2000).

References

Arimoto S (1996) *Control Theory of Non-linear Mechanical Systems: A Passivity-based and Circuit-theoretic Approach*. New York: Oxford University Press.

Baker S and Matthews I (2004) Lucas–Kanade 20 years on: A unifying framework. *International Journal of Computer Vision* 56(3): 221–255.

Berenson D (2013) Manipulation of deformable objects without modeling and simulating deformation. In: *Proceedings of the IEEE/RSJ International conference on intelligent robots and systems*, Tokyo, 3–18 November 2013, pp. 1–8.

Bradski G (2000) The OpenCV Library. *Dr. Dobb's Journal of Software Tools* 25(11): 120, 122–125.

Broyden CG (1965) A class of methods for solving nonlinear simultaneous equations. *Mathematics of Computation* 19: 577–593.

Chaumette F (2004) Image moments: a general and useful set of features for visual servoing. *IEEE Transactions on Robotics* 20(4): 713–723.

Chaumette F and Hutchinson S (2006) Visual servo control. Part I: Basic approaches. *IEEE Robotics and Automation Magazine* 13(4): 82–90.

Craig JJ (1989) *Introduction to Robotics: Mechanics and Control*. 2nd edition. Boston, MA: Addison-Wesley Longman Publishing Co.

Cusumano-Towner M, Singh A, Miller S, et al. (2011) Bringing clothing into desired configurations with limited perception. In: *Proceedings of IEEE International conference on robotics and automation*, Shanghai, 9–13 May 2011, pp. 3893–3900.

Das J and Sarkar N (2011) Autonomous shape control of a deformable object by multiple manipulators. *Journal of Intelligent and Robotic Systems* 62: 3–27.

Forsyth DA and Ponce J (2002) *Computer Vision: A Modern Approach*. 1st edition. Englewood Cliffs, NJ: Prentice Hall.

Gerum P (2004) *Xenomai - Implementing a RTOS emulation framework on GNU/Linux*. White paper. Available at: http://www.xenomai.org/documentation/branches/v2.3.x/pdf/xenomai.pdf.

Hartley RI and Zisserman A (2004) *Multiple View Geometry in Computer Vision*. 2nd edition. Cambridge: Cambridge University Press.

Henrich D and Worn H (eds) (2000) *Robot Manipulation of Deformable Objects*. (Advanced manufacturing). New York: Springer-Verlag.

Higashimori M, Yoshimoto K and Kaneko M (2010) Active shaping of an unknown rheological object based on deformation decomposition into elasticity and plasticity. In: *Proceedings of the IEEE International conference on robotics and automation*, Anchorage, AK, USA, 3–8 May 2011, pp. 5120–5126.

Hirai S and Wada T (2000) Indirect simultaneous positioning of deformable objects with multi-pinching fingers based on an uncertain model. *Robotica* 18(1): 3–11.

Hogan N (1985) Impedance control: An approach to manipulation, parts I–III. *ASME Journal of Dynamic Systems, Measurement, and Control* 107: 1–24.

Hokayem PF and Spong MW (2006) Bilateral teleoperation: An historical survey. *Automatica* 42(12): 2035–2057.

Hosoda K and Asada M (1994) Versatile visual servoing without knowledge of true Jacobian. In: *Proceedings of the IEEE International conference on Intelligent robots and systems*, Sendai, Japan, 28 September–2 October 2004, Vol. 1, pp. 186–193.

Hutchinson S, Hager G and Corke P (1996) A tutorial on visual servo control. *IEEE Transactions on Robotics Automation* 12(5): 651–670.

Jagersand M, Fuentes O and Nelson R (1997) Experimental evaluation of uncalibrated visual servoing for precision manipulation. In: *Proceedings of the IEEE International conference on robotics and automation*, Albuquerque, NM, USA, 20–25 April 1997, Vol. 4, pp. 2874–2880.

Kimberling C (1998) *Triangle Centers and Central Triangles: By Clark Kimberling*. (Congressus Numerantum). Winnipeg, Canada: Utilitas Mathematica Publishing.

Kimura M, Sugiyama Y, Tomokuni S, et al. (2003) Constructing rheologically deformable virtual objects. In: *Proceedings of the IEEE International conference on robotics and automation*, Taipei, Taiwan, 14–19 September 2003, Vol. 3, pp. 3737–3743.

Kinio S and Patriciu A (2012) A comparative study of Hinf and PID control for indirect deformable object manipulation. In: *IEEE International conference on robotics and biomimetics*, St Paul, MN, USA, 14–18 May 2012, pp. 414–420.

Lee CSG and Ziegler M (1984) Geometric approach in solving inverse kinematics of puma robots. *IEEE Transactions on Aerospace and Electronic Systems* AES-20(6): 695–706.

Liu YH and Sun D (2000) Stabilizing a flexible beam handled by two manipulators via PD feedback. *IEEE Transactions Automatic Control* 45(11): 2159–2164.

Liu YH, Wang H, Chen W, et al. (2013) Adaptive visual servoing using common image features with unknown geometric parameters. *Automatica* 49(8): 2453–2460.

Liu YH, Wang H, Wang C, et al. (2006) Uncalibrated visual servoing of robots using a depth-independent interaction matrix. *IEEE Transactions on Robotics* 22(4): 804–817.

Mallapragada V, Sarkar N and Podder T (2011) Toward a robot-assisted breast intervention system. *IEEE/ASME Transactions on Mechatronics* 16(6): 1011–1020.

Marsden JE and Ratiu T (1994) *Introduction to Mechanics and Symmetry* (Texts in Applied Mathematics). New York: Springer-Verlag.

Mills JK and Ing JGL (1996) Dynamic modeling and control of a multi-robot system for assembly of flexible payloads with applications to automotive body assembly. *Journal of Robotic Systems* 13(12): 817–836.

Murray RM, Sastry SS and Zexiang L (1994) *A Mathematical Introduction to Robotic Manipulation*. 1st edition. Boca Raton, FL: CRC Press.

Nakamura Y (1991) *Advanced Robotics: Redundancy and Optimization*. 1st edition. Boston, MA: Addison-Wesley Longman Co.

Navarro-Alarcon D and Liu YH (2013) Uncalibrated vision-based deformation control of compliant objects with online estimation of the Jacobian matrix. In: *Proceedings of the IEEE International conference on intelligent robots and systems*, Tokyo, 3–8 November 2013, pp. 4977–4982.

Navarro-Alarcon D, Liu YH, Romero JG, et al. (2013) Model-free visually servoed deformation control of elastic objects by robot manipulators. *IEEE Transactions on Robotics* 26(6): 1457–1468.

Ogden R (1997) *Non-Linear Elastic Deformations* (Dover Civil and Mechanical Engineering Series). Mineola, NY: Dover Publications.

Okamura A, Smaby N and Cutkosky M (2000) An overview of dexterous manipulation. In: *Proceedings of the IEEE International conference on robotics and automation*, San Francisco, CA, USA, 24–28 April 2000, Vol. 1, pp. 255–262.

Ortega R, van der Schaft A, Mareels I, et al. (2001) Putting energy back in control. *IEEE Control Systems Magazine* 21(2): 18–33.

Park E and Mills J (2005) Static shape and vibration control of flexible payloads with applications to robotic assembly. *IEEE/ASME Transactions on Mechatronics* 10(6): 675–687.

Pertin F and Bonnet des Tuves JM (2004) Real time robot controller abstraction layer. In: *Proceedings of the International symposium on robotics*, p. 71.

Piepmeier J, McMurray G and Lipkin H (2004) Uncalibrated dynamic visual servoing. *IEEE Transactions on Robotics Automation* 20(1): 143–147.

Romero JG, Donaire A and Ortega R (2013) Robust energy shaping control of mechanical systems. *Systems and Control Letters* 62(9): 770–780.

Saha M and Isto P(2007) Manipulation planning for deformable linear objects. *IEEE Transactions on Robotics* 23(6): 1141–1150.

Shen Y, Sun D, Liu YH, et al. (2003) Asymptotic trajectory tracking of manipulators using uncalibrated visual feedback. *IEEE/ASME Transactions on Mechatronics* 8(1): 87–98.

Siciliano B (1990) Kinematic control of redundant robot manipulators: A tutorial. *Journal of Intelligent and Robotic Systems* 3(3): 201–212.

Slotine JJ and Li W (1991) *Applied Nonlinear Control*. 1st edition. Englewood Cliffs, NJ: Prentice Hall.

Smolen J and Patriciu A (2009) Deformation planning for robotic soft tissue manipulation. In: *Proceedings of the International conference on advances in computer-human interactions*, Cancun, Mexico, pp. 199–204.

Sun D and Liu YH (2001) Position and force tracking of a two-manipulator system manipulating a flexible beam. *Journal of Robotic Systems* 18(4): 197–212.

Sun D, Mills JK and Liu Y (1999) Position control of robot manipulators manipulating a flexible payload. *International Journal of Robotics Research* 18(3): 319–332.

Tokumoto S and Hirai S (2002) Deformation control of rheological food dough using a forming process model. In: *Proceedings of the IEEE International conference on robotics and automation*, Washington DC, USA, 11–15 May 2002, Vol. 2, pp. 1457–1464.

Tokumoto S, Hirai S and Tanaka H (2001) Constructing virtual rheological objects. In: *Proceedings of the world multiconference on systemics, cybernetics and informatics*, Orlando, FL, USA, pp. 106–111.

Torabi M, Hauser K, Alterovitz R, et al. (2009) Guiding medical needles using single-point tissue manipulation. In: *Proceedings of the IEEE International conference on robotics and automation*, Kobe, Japan, 12–17 May 2009, pp. 2705–2710.

Vidyasagar M (1993) *Nonlinear Systems Analysis*. 2nd edition. Englewood Cliffs, NJ: Prentice Hall.

Wada T, Hirai S, Kawamura S, et al. (2001) Robust manipulation of deformable objects by a simple PID feedback. In: *Proceedings of the IEEE International conference on robotics and automation*, Seoul, South Korea, 21–26 May 2001, Vol. 1, pp. 85–90.

Wakamatsu H and Hirai S (2004) Static modeling of linear object deformation based on differential geometry. *International Journal of Robotics Research* 23(3): 293–311.

Wang H, Liu YH and Zhou D (2008) Adaptive visual servoing using point and line features with an uncalibrated eye-in-hand camera. *IEEE Transactions on Robotics* 24(4): 843–857.

Wang Z and Hirai S (2011) Modeling and estimation of rheological properties of food products for manufacturing simulations. *Journal of Food Engineering* 102(2): 136–144.

Whitney D (1969) Resolved motion rate control of manipulators and human prostheses. *IEEE Transactions on Man Machine Systems* 10(2): 47–53.

Yoshikawa T (2010) Multifingered robot hands: Control for grasping and manipulation. *Annual Reviews in Control* 34(2): 199–208.

Yue S and Henrich D (2005) Manipulating deformable linear objects: Sensor-based skills of adjustment motions for vibration reduction. *Journal of Robotic Systems* 22(2): 67–85.

Appendix: Index to Multimedia Extension

The multimedia extension page is found at: http://www.ijrr.org

Table of Multimedia Extension

Extension	Media type	Description
1	Video	Three examples of visually servoed deformation experiments. The first two tasks demonstrate the performance of the flow-based controller; the last task demonstrates the performance of the adaptive controller.