2013 IEEE/RSJ International Conference on
Intelligent Robots and Systems (IROS)
November 3-7, 2013. Tokyo, Japan

Uncalibrated Vision-based Deformation Control of Compliant Objects with Online Estimation of the Jacobian Matrix

David Navarro-Alarcon and Yun-hui Liu

***Abstract*— In this paper, we propose a new vision-based controller to actively deform an unknown elastic object. Note that most deformation controllers in the literature require a-priori knowledge of the object's deformation properties. In contrast to this trend, we present a new Lyapunov-based method that online estimates the unknown deformation Jacobian matrix, avoiding any model identification or calibration steps. To achieve the desired object's deformation, we derive an innovative dynamic-state feedback velocity control law using the passivity-based framework. We present a detailed experimental study to validate the feasibility of our deformation controller.**

I. Introduction

The automatic deformation control of compliant objects refers to applications where a mechanical system (typically a manipulator) actively shapes a soft object into a desired form [1]. In recent years, this compliant manipulation problem is receiving considerable attention from the research community because of its economically important applications in, for example, robotic surgery [2], home-robots [3], and food industry [4]. Despite these research efforts, one of the principal issues that still hinders the successful implementation of these tasks, is the difficulty to identify/estimate the deformation properties of soft materials.

The active deformation control problem has been studied by some research groups. The modelling and parameter estimation of deformable objects is addressed in [1], [5], [6]. In [7], a two-phase control method (parameter identification, then model-based control) is proposed to actively modify the shape of a rheological material. In [8], the problem of the simultaneous motion and one-dimensional deformation (i.e. compression) of soft materials is addressed. A model-based control method to indirectly position deformable points by robotic fingers is reported in [9]. This indirect positioning problem is also addressed in [10], [11]. In [12], a controller to shape a known flexible body (modelled with finite element method) by multiple manipulators is reported.

In one way or another, most control methods in the literature (including the ones we mention above) require a-priori knowledge of the object's deformation properties (e.g. visual/kinematic relations, dynamic models, or elastic/rheological parameters). Recently, we have proposed in [13] an entirely model-free controller that uses the Broyden update rule [14] to estimate (on the fly) the deformation Jacobian matrix. This numerical technique does not require prior identification of the object's properties, however, it only provides "numerical snapshots" of the unknown deformation matrix. Then, it is clear the need to analytically develop a new estimation method with guaranteed stability.

To contribute to this problem, in this paper we propose a new vision-based control method to actively deform an *unknown* elastic object. Our controller is uncalibrated since it does not require a-priori knowledge of the object's deformation properties or the camera's setup. For that, our new method adaptively estimates the unknown deformation Jacobian matrix; we prove the stability of this algorithm with Lyapunov theory. To smoothly deform the object, we propose an innovative dynamic-state feedback velocity controller. We present an experimental study to validate the feasibility of this control method.

The rest of this paper is organized as follows. In Section II, we derive the mathematical models. Section III introduces the problem. In Section IV, we present our new deformation controller. Section V reports the experimental results. Finally, we give final conclusions and future work in Section VI.

II. Mathematical modelling

A. Manipulator model

For the serial robot manipulator under consideration, we denote the vectors of joint and end-effector displacements by $\mathbf{q}(t) \in \mathbb{R}^g$ and $\mathbf{x}(t) \in \mathbb{R}^3$ (where $g \geq 3$), respectively. The differential relation between the joint and end-effector velocities is given by the expression

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

where $\frac{\partial \mathbf{x}}{\partial \mathbf{q}}(\mathbf{q}(t)) \in \mathbb{R}^{3\times g}$ represents the Jacobian matrix of the manipulator, which we assume to be exactly known.

We denote the manipulator's control input by $\boldsymbol{\omega}(t) \in \mathbb{R}^g$. Physically, this variable represents the *bounded* angular velocity input to the joints. Without loss of generality, we assume that the control architecture of this kinematically-controlled system allows to *exactly* set the joint's angular velocity such that $\boldsymbol{\omega}(t) \equiv \dot{\mathbf{q}}(t)$ is always satisfied.

B. Deformation model

In this paper, we address the automatic shaping problem of compliant objects that exhibit elastic deformations only (i.e. we do not address interactions with rheological materials). To actively modify the object's shape, we consider that the manipulator's end-effector physically interacts with the object in a fully constrained manner. This fully constrained situation refers to the case where the motion (in any direction) of

Both authors are with the Department of Mechanical and Automation Engineering of the Chinese University of Hong Kong, Shatin NT, Hong Kong SAR. dnavarro,yhliu[at]mae.cuhk.edu.hk

This work is supported in part by Hong Kong RGC under grant 415011, the Hong Kong ITF under the project ITS/475/09.

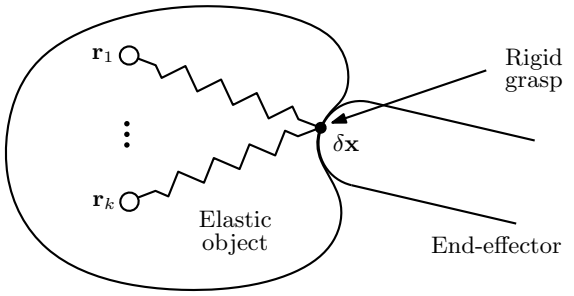


Fig. 1. Conceptual representation of the deformation model, where the end-effector is virtually connected by spatial springs with the k points.

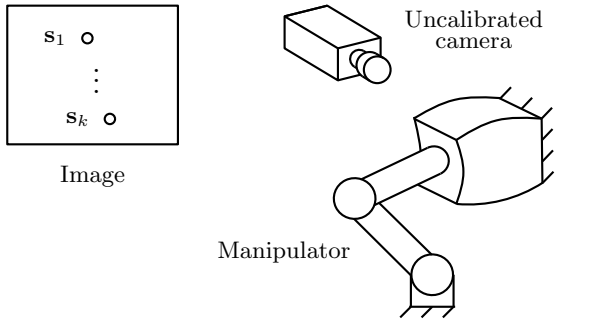


Fig. 2. Conceptual representation of a fixed uncalibrated camera providing the visual feedback of k visual points.

the end-effector results in a proportional deformation of the body. It models, for example, tasks where the manipulator *rigidly* grasps a soft material.

To characterise the deformation of the object, in our method we make use k Cartesian points of interest conveniently located on the object's surface. The i-th Cartesian point is denoted by the vector $\mathbf{r}_i(t) \in \mathbb{R}^3$. In our approach, we *locally* approximate the coordinates of each point by

$$\mathbf{r}_i(t) = \mathbf{C}_i \delta\mathbf{x}(t), \tag{2}$$

where we introduce the vector $\delta\mathbf{x}(t) = \mathbf{x}(t) - \overline{\mathbf{x}} \in \mathbb{R}^3$ to model the relative end-effector displacement, for $\overline{\mathbf{x}} \in \mathbb{R}^3$ as a constant bias vector. The unknown constant matrix $\mathbf{C}_i \in \mathbb{R}^{3\times 3}$, which we assume to be full rank, contains the deformation parameters of the point $\mathbf{r}_i(t)$. In total, there are $k \times 9$ independent deformation parameters relating the displacement of the points $\mathbf{r}_i(t)$ with $\mathbf{x}(t)$. To simplify notation, we define the following total position vector

$$\mathbf{r}(t) = \begin{bmatrix}\mathbf{r}_1^\top(t) & \cdots & \mathbf{r}_k^\top(t)\end{bmatrix}^\top \in \mathbb{R}^{3k}. \tag{3}$$

See Fig. 1 for a conceptual representation of this setup.

C. Camera model

To measure the object's deformation, in our approach we make use of an *uncalibrated* fixed camera that provides the visual feedback of the scene. We define the visual feedback of the Cartesian point $\mathbf{r}_i(t)$ as follows

$$\mathbf{s}_i(t) = \begin{bmatrix}u_i(t) & v_i(t)\end{bmatrix}^\top \in \mathbb{R}^2, \tag{4}$$

where the coordinates $u_i(t), v_i(t) \in \mathbb{R}$ are measured in pixel units. To simplify notation, we introduce the following total vector of visual feedback

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

A conceptual representation of this setup is given in Fig. 2.

Instrumental for the online Jacobian matrix estimator that we present in Section IV, is to derive a simple and linearly parametrisable visual projection model. To this end, we assume that, *locally* around the desired operating point, the visual feedback point $\mathbf{s}_i(t)$ satisfies the affine projection model [15]

$$\mathbf{s}_i(t) = \mathbf{M}\mathbf{r}_i(t) + \mathbf{b} \in \mathbb{R}^2, \tag{6}$$

where the constant matrix $\mathbf{M} \in \mathbb{R}^{2\times 3}$ and vector $\mathbf{b} \in \mathbb{R}^2$ are unknown. As we show later on in Section IV, the use of this affine projection model does not impose severe constraints to the iterative Jacobian matrix estimator that we develop.

Based on the deformation and projection models (2) and (6), we now derive the following differential relation between the manipulator's motion and the total visual flow

$$\dot{\mathbf{s}}(t) = \mathbf{L}\dot{\mathbf{x}}(t), \tag{7}$$

for a constant matrix $\mathbf{L} \in \mathbb{R}^{2k\times 3}$ defined as

$$\mathbf{L} = \begin{bmatrix}\mathbf{M}\mathbf{C}_1 \\ \vdots \\ \mathbf{M}\mathbf{C}_k\end{bmatrix}, \tag{8}$$

which groups the deformation parameters of each point $\mathbf{r}_i(t)$, and the camera projection parameters (with the exception of the "bias" $\mathbf{b}$). We list the elements of $\mathbf{L}$ in the vector of unknown parameters

$$\boldsymbol{\theta} = \begin{bmatrix}L_{1,1} & L_{1,2} & \cdots & L_{2k,2} & L_{2k,3}\end{bmatrix}^\top \in \mathbb{R}^m, \tag{9}$$

where $m = 6k \in \mathbb{R}$, and $L_{i,j} \in \mathbb{R}$ represents an element at the i-th row and j-th column of the matrix $\mathbf{L}$.

III. Problem statement

To quantify the deformation of the object, in this section we construct as a known function of the total visual feedback, the following *deformation feature* vector

$$\mathbf{y}(t) = \mathbf{y}(\mathbf{s}(t)) \in \mathbb{R}^3. \tag{10}$$

Similar to standard visual servo controllers (such as [16], [17]), the independent coordinates of $\mathbf{y}(t)$ are composed of, for example, point displacements, angles of a lines, centroids, midpoints, curvatures of contours, line features, to name a few cases (see [13] for detailed definitions of several deformation features).

We differentiate (10) with respect to time to obtain the expression

$$\dot{\mathbf{y}}(t) = \mathbf{J}(\mathbf{x}(t))\dot{\mathbf{x}}(t), \tag{11}$$

where $\mathbf{J}(\mathbf{x}(t)) \in \mathbb{R}^{3\times 3}$ represents the *deformation Jacobian matrix* of the elastic object, which we define by

$$\mathbf{J}(\mathbf{x}(t)) = \frac{\partial \mathbf{y}}{\partial \mathbf{s}}(\mathbf{s}(t))\mathbf{L}. \tag{12}$$

Note that the matrix $\mathbf{J}(\mathbf{x}(t))$ is unknown since it is constructed with the unknown parameters $\boldsymbol{\theta}$.

For ease of presentation, we make a change in the bounded velocity control input, such that

$$\dot{\mathbf{x}}(t) = \mathbf{v}(t) \in \mathbb{R}^3, \tag{13}$$

represents the new control input (i.e. the end-effector velocity) to the deformation plant (11). Since the manipulator's kinematics are exactly known, therefore, we can easily design a joint kinematic controller $\boldsymbol{\omega}(t)$ which moves the end-effector at a desired velocity $\dot{\mathbf{x}}(t)$.

Problem. Given a desired constant deformation feature vector $\mathbf{y}_d \in \mathbb{R}^3$, design an uncalibrated (i.e. with no a-priori knowledge of the deformation and projection parameters) velocity control law $\mathbf{v}(t)$ which asymptotically minimises the deformation error $\Delta\mathbf{y}(t) = \mathbf{y}(t) - \mathbf{y}_d \in \mathbb{R}^3$.

IV. Controller design

A. Online estimation of $\mathbf{J}(\mathbf{x}(t))$

To minimise the deformation error $\Delta\mathbf{y}(t)$, we must first estimate how the velocity control input actively changes the shape of the object. In other words, we must identify the unknown deformation mapping $\dot{\mathbf{x}}(t) \mapsto \dot{\mathbf{y}}(t)$. This directional information is provided by the object's deformation Jacobian matrix.

For ease of presentation of our online estimator, let us introduce the vector of adaptive parameters $\widehat{\boldsymbol{\theta}}(t) \in \mathbb{R}^m$ (whose explicit update rule is presented shortly). With this adaptive vector available and with the real-time visual measurement of the deformation flow $\dot{\mathbf{y}}(t)$[1], we compute the following *flow estimation error*

$$\mathbf{e}(t) = \mathbf{z}(t) - \dot{\mathbf{y}}(t) \in \mathbb{R}^3, \tag{14}$$

for an estimated flow vector $\mathbf{z}(t) \in \mathbb{R}^3$ defined as

$$\mathbf{z}(t) = \frac{\partial \mathbf{y}}{\partial \mathbf{s}}(\mathbf{s}(t))\widehat{\mathbf{L}}(t)\dot{\mathbf{x}}(t), \tag{15}$$

where the matrix $\widehat{\mathbf{L}}(t) \in \mathbb{R}^{2k\times 3}$ is constructed with the vector of adaptive parameters $\widehat{\boldsymbol{\theta}}(t)$. By substituting (11) and (15) into (14), we obtain a linearly parametrisable expression for the flow error [18]

$$\begin{aligned}\mathbf{e}(t) &= \frac{\partial \mathbf{y}}{\partial \mathbf{s}}(\mathbf{s}(t))\left(\widehat{\mathbf{L}}(t) - \mathbf{L}\right)\dot{\mathbf{x}}(t),\\ &= \mathbf{W}(\mathbf{s}(t), \dot{\mathbf{x}}(t))\Delta\boldsymbol{\theta}(t),\end{aligned} \tag{16}$$

where $\Delta\boldsymbol{\theta}(t) = \widehat{\boldsymbol{\theta}}(t) - \boldsymbol{\theta} \in \mathbb{R}^m$ represents the parameters' estimation error, and $\mathbf{W}(\mathbf{s}(t), \dot{\mathbf{x}}(t)) \in \mathbb{R}^{3\times m}$ represents a *known* regression matrix whose computation does not dependent on the true parameters. Considering the definition of $\boldsymbol{\theta}$ in (9), we compute the matrix $\mathbf{W}(\mathbf{s}(t), \dot{\mathbf{x}}(t))$ with the simple expression

$$\mathbf{W}(t) = \frac{\partial \mathbf{y}}{\partial \mathbf{s}}(\mathbf{s}(t))\begin{bmatrix} \dot{\mathbf{x}}^\top(t) & \mathbf{0}_{1\times 3} & \cdots & \mathbf{0}_{1\times 3} \\ \mathbf{0}_{1\times 3} & \dot{\mathbf{x}}^\top(t) & \cdots & \mathbf{0}_{1\times 3} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{0}_{1\times 3} & \mathbf{0}_{1\times 3} & \cdots & \dot{\mathbf{x}}^\top(t) \end{bmatrix}. \tag{17}$$

[1]Note that low-pass filtering may be needed in order to use this signal.

Proposition 1: For smooth, slow, and continuous deformation tasks, the parameter adaptation rule

$$\dot{\widehat{\boldsymbol{\theta}}}(t) = -\boldsymbol{\Gamma}\mathbf{W}^\top(\mathbf{s}(t), \dot{\mathbf{x}}(t))\mathbf{K}_\theta\mathbf{e}(t), \tag{18}$$

where $\mathbf{K}_\theta > 0 \in \mathbb{R}^{3\times 3}$ and $\boldsymbol{\Gamma} > 0 \in \mathbb{R}^{m\times m}$ represent positive-definite symmetric tuning matrices, guarantees the following two conditions:

(i) A numerically stable adaptation of the parameters $\widehat{\boldsymbol{\theta}}(t)$.
(ii) The asymptotic minimisation of the flow error $\mathbf{e}(t)$.

Proof: We present a sketch of this proof.

(i) Considering (16), the adaptation rule (18) can be equivalently re-written as

$$\dot{\widehat{\boldsymbol{\theta}}}(t) = -\boldsymbol{\Gamma}\mathbf{W}^\top(t)\mathbf{K}_\theta\mathbf{W}(t)\Delta\boldsymbol{\theta}(t). \tag{19}$$

To prove the stability of (18), we propose the positive-definite Lyapunov function

$$\begin{aligned}\mathcal{Q}(t) &= \frac{1}{2}\Delta\boldsymbol{\theta}^\top(t)\boldsymbol{\Gamma}^{-1}\Delta\boldsymbol{\theta}(t) \in \mathbb{R},\\ &> 0,\end{aligned} \tag{20}$$

whose time derivative along trajectories of (18) satisfies

$$\begin{aligned}\dot{\mathcal{Q}}(t) &= -\Delta\boldsymbol{\theta}^\top(t)\mathbf{W}^\top(t)\mathbf{K}_\theta\mathbf{W}(t)\Delta\boldsymbol{\theta}(t),\\ &= -\mathbf{e}^\top(t)\mathbf{K}_\theta\mathbf{e}(t),\\ &\leq 0.\end{aligned} \tag{21}$$

This expression shows that $\dot{\mathcal{Q}}(t)$ is negative semidefinite. From the Lyapunov's direct method [19], we prove that the adaptation of the parameters is stable, i.e., the vector $\widehat{\boldsymbol{\theta}}(t)$ is bounded.

(ii) From (21), we know that $\mathcal{Q}(t)$ is monotonically decreasing, i.e. $\mathcal{Q}(t_1) \geq \mathcal{Q}(t_2)\ \forall t_1 \leq t_2$. Note that since we assume smooth and slow (bounded) input/output motion, simple computations show that $\ddot{\mathcal{Q}}(t)$ is also bounded. These conditions imply that as $t \to \infty$ then $\mathcal{Q}(t) \to c$ and $\dot{\mathcal{Q}}(t) \to 0$, for a scalar $c \geq 0 \in \mathbb{R}$. From (21), we see that $\dot{\mathcal{Q}}(t) \equiv 0 \Rightarrow \mathbf{e}(t) \equiv \mathbf{0}_{3\times 1}$. Therefore, we conclude that as $t \to \infty$ then $\mathbf{e}(t) \to \mathbf{0}_{3\times 1}$ [19].

■

In our new method, we adaptively compute an estimation of the unknown deformation Jacobian matrix by

$$\widehat{\mathbf{J}}(t) = \frac{\partial \mathbf{y}}{\partial \mathbf{s}}(\mathbf{s}(t))\widehat{\mathbf{L}}(t) \in \mathbb{R}^{3\times 3}. \tag{22}$$

This method contrast with our previous purely numeric approach [13] in that we online compute a vector of adaptive parameters $\widehat{\boldsymbol{\theta}}(t)$, and not iteratively obtain "numerical snapshots" of the whole matrix $\mathbf{J}(\mathbf{x}(t))$ as in [13].

Remark 1: Note that the $m \times m$ "dissipation-like" matrix $\mathbf{W}^\top(t)\mathbf{K}_\theta\mathbf{W}(t)$ in (21) is not full rank. Therefore, we can not guarantee that (18) asymptotically estimates the m-dimensional (recall $m = 6k$) vector of true parameters $\boldsymbol{\theta}$. In our method, the objective of the adaptation rule (18) is to construct a 3×3 matrix $\widehat{\mathbf{J}}(t)$ that closely approximates (with no prior model calibration) the measured deformation flow

$$\dot{\mathbf{y}}(t) \approx \widehat{\mathbf{J}}(t)\dot{\mathbf{x}}(t). \tag{23}$$

Remark 2: Note that the online estimation algorithm requires the real-time measurement of the visual flow and the manipulator's end-effector velocity. For slow tasks, these quantities can be accurately obtained by using simple numerical differentiation methods along with low-pass filtering of the obtained flow signal.

Remark 3: The affine projection model (6) locally approximates the visual feedback points $\mathbf{s}_i(t)$ with a *constant* vector of parameters $\boldsymbol{\theta}$. Note that in this paper we only consider slow deformation tasks, therefore we can still use the same affine model to approximate $\mathbf{s}_i(t)$ for wider elastic deformations. In this situation, the parameters that approximate the visual feedback are expected to change slowly, thus $\frac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{\theta} \approx \mathbf{0}_{m\times 1}$ is a reasonable assumption for these types of motions.

B. Velocity controller

To control the behaviour of the system (11), we propose the dynamic-state feedback control law [13]

$$\mathbf{v}(t) = \widehat{\mathbf{J}}^{-1}(t)\mathbf{p}(t), \tag{24}$$

where the vector $\mathbf{p}(t) \in \mathbb{R}^3$ represents a smooth numerical state, which we compute as

$$\dot{\mathbf{p}}(t) = -\frac{\partial \mathcal{U}}{\partial \mathbf{y}}^{\top}(\Delta\mathbf{y}(t)) - \mathbf{K}_p\mathbf{p}(t), \tag{25}$$

for $\mathcal{U}(\Delta\mathbf{y}(t)) > 0 \in \mathbb{R}$ as a positive-definite potential energy functional designed with a unique equilibrium point at $\Delta\mathbf{y}(t) = \mathbf{y}(t) - \mathbf{y}_d = \mathbf{0}_{3\times 1}$, and $\mathbf{K}_p = \mathbf{K}_p^{\top} > 0 \in \mathbb{R}^{3\times 3}$ as a damping-like feedback matrix.

Proposition 2: For an exact estimation of the deformation Jacobian matrix (i.e. for $\mathbf{J}(\mathbf{x}(t)) = \widehat{\mathbf{J}}(t)$), the velocity control input (24) enforces asymptotic stability of the deformation error vector $\Delta\mathbf{y}(t)$.

Proof: Substitution of (24) into (11) enforces the closed-loop dynamical system

$$\begin{bmatrix}\dot{\mathbf{y}}(t)\\ \dot{\mathbf{p}}(t)\end{bmatrix} = \begin{bmatrix}\mathbf{0}_{3\times 3} & \mathbf{I}_{3\times 3}\\ -\mathbf{I}_{3\times 3} & -\mathbf{K}_p\end{bmatrix}\begin{bmatrix}\frac{\partial \mathcal{H}}{\partial \mathbf{y}}^{\top}(t)\\ \frac{\partial \mathcal{H}}{\partial \mathbf{p}}^{\top}(t)\end{bmatrix}, \tag{26}$$

with a Hamiltonian-like function

$$\mathcal{H}(\Delta\mathbf{y}(t), \mathbf{p}(t)) = \mathcal{U}(\Delta\mathbf{y}(t)) + \frac{1}{2}\mathbf{p}^{\top}(t)\mathbf{p}(t) \in \mathbb{R}, \\ > 0. \tag{27}$$

This positive-definite scalar qualifies as a Lyapunov function for the dynamical system (26), since its time-derivative satisfies

$$\dot{\mathcal{H}}(\Delta\mathbf{y}(t), \mathbf{p}(t)) \leq 0. \tag{28}$$

To prove asymptotic convergence of the deformation error $\Delta\mathbf{y}(t)$, we invoke the Krasovskii–laSalle principle [19]. ■

Remark 4: The velocity control input computed with (24) provides smooth trajectories to the robot manipulator. Note that the dynamic-state feedback control action $\mathbf{p}(t)$ (which is basically a numerical integrator driven by the potential error and damped by its own feedback) helps to remove noise from the visual measurements.

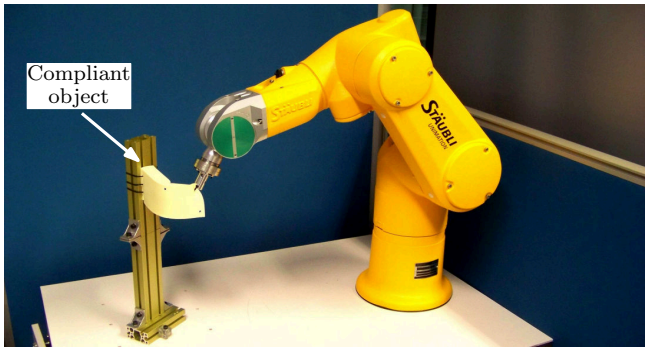


Fig. 3. The experimental setup, where the Stäbli robot manipulator physically interacts with an unknown compliant object.

Remark 5: We can interpret the closed-loop dynamics (26) as a non-conservative Hamiltonian system [20]. This way, the desired closed-loop deformation behaviour is defined by the potential energy function $\mathcal{U}(\Delta\mathbf{y}(t))$ and the dissipation matrix $\mathbf{K}_p$. We can physical interpret the state variables $\mathbf{y}(t)$ and $\mathbf{p}(t)$ as the canonical (visual) displacements and momenta, respectively.

Algorithm 1 presents the implementation of our uncalibrated deformation controller, where $small\_error > 0 \in \mathbb{R}$ represents the minimum admissible error.

Algorithm 1 Uncalibrated deformation controller

1: **repeat** ▷ Control loop
2: Measure signals $\mathbf{x}(t)$ and $\mathbf{s}(t)$
3: Adapt parameters $\frac{\mathrm{d}}{\mathrm{d}t}\widehat{\boldsymbol{\theta}}(t) \leftarrow$ (18)
4: Compute state $\frac{\mathrm{d}}{\mathrm{d}t}\mathbf{p}(t) \leftarrow$ (25)
5: Compute controller $\mathbf{v}(t) \leftarrow$ (24)
6: Solve $\mathbf{v}(t)$ for $\boldsymbol{\omega}(t)$ and command motion
7: **until** Error $\|\Delta\mathbf{y}(t)\| < small\_error$

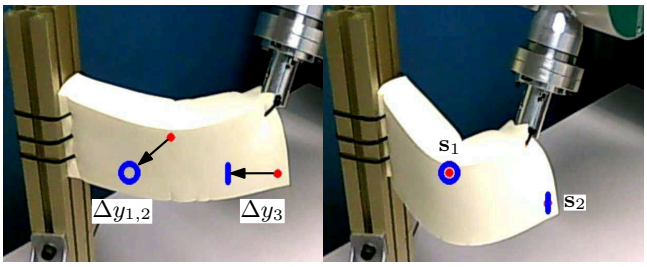
V. Experimental validation

A. Setup

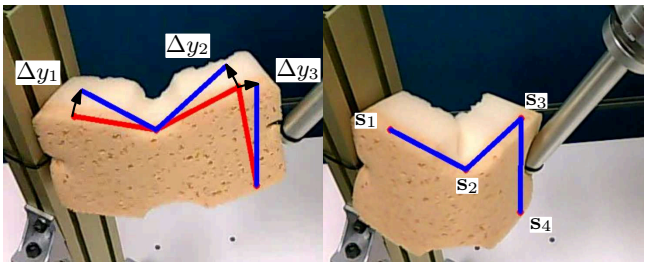
The robotic system used for our experimental study is a 6-DOF TX-60 Stäubli robot manipulator (see Fig. 3). This manipulator has an open architecture controller running the real-time operative system VxWorks. To program the motion of its joints, our system counts the *Low-Level Interface* that allows to explicitly set the angular velocity on each of the joints in real-time (see [13] for details about our real-time control implementation). Visual feedback is acquired with a USB Logitech camera; we use the OpenCV Lucas–Kanade algorithm to track the visual points during the experiments[2].

In our experimental study, the end-effector is rigidly attached to common household cleaning sponges (whose deformation properties are fairly approximated with an elastic deformation model). In our formulation, we only consider the control of the end-effector 3-DOF Cartesian displacements,

[2]For that, we place artificial markers on the object's surface.



(a) Point-based deformation features.



(b) Angle-based deformation features.

Fig. 4. Snapshots of the initial and final configurations of the performed 3-DOF deformation experiments.

therefore, we perform these experiments with a constant orientation of the manipulator's end-effector.

B. Case of study

We test the proposed control method with two different 3-DOF deformation tasks. The first task is defined by constructing the vector $\mathbf{y}(t) = [y_1(t), y_2(t), y_3(t)]^\top \in \mathbb{R}^3$ in terms of the following pixel displacement coordinates

$$\mathbf{y}(\mathbf{s}(t)) = [\mathbf{s}_1^\top(t), u_2(t)]^\top, \tag{29}$$

where $u_2(t)$ represents the *horizontal* displacement of $\mathbf{s}_2(t)$. The second task is defined by constructing the coordinates of $\mathbf{y}(t)$ in terms of the following angular displacements

$$y_i(t) = \arccos(\mathbf{o}_i \cdot \mathbf{l}_i(t)) \in \mathbb{R}, \quad i = 1, 2, 3 \tag{30}$$

for an arbitrary unit reference vector $\mathbf{o}_i \in \mathbb{R}^2$, and a state-dependent vector $\mathbf{l}_i(t) = \frac{\mathbf{s}_i(t) - \mathbf{s}_{i+1}(t)}{\|\mathbf{s}_i(t) - \mathbf{s}_{i+i}(t)\|} \in \mathbb{R}^2$. See Fig. 4 for a graphical representation of these deformation features.

In this experimental study, we implement the potential control action

$$\frac{\partial \mathcal{U}}{\partial \mathbf{y}}^\top (\Delta\mathbf{y}(t)) = \mathbf{sat}(3\Delta\mathbf{y}(t)), \tag{31}$$

where $\mathbf{sat}(\cdot) : \mathbb{R}^3 \mapsto \mathbb{R}^3$ is a vectorial saturation function satisfying $-\boldsymbol{\alpha} < \mathbf{sat}(\cdot) < \boldsymbol{\alpha}$, for $\boldsymbol{\alpha} > 0 \in \mathbb{R}^3$ as a constant bound vector. We respectively use $\boldsymbol{\alpha} = [7, 7, 7]^\top$ pixel and $\boldsymbol{\alpha} = [0.1, 0.1, 0.1]^\top$ radian for our two cases of study. For both cases, the dynamic-state feedback action $\mathbf{p}(t)$ is computed with a damping matrix $\mathbf{K}_p = 5\mathbf{I}_{3\times3}$.

To implement the proposed online estimator (18), we remove noise from the signals $\dot{\mathbf{y}}(t)$ and $\dot{\mathbf{x}}(t)$ by using a first-order low-pass filter. We compute $\widehat{\mathbf{J}}(t)$ with the tuning gain matrices $\boldsymbol{\Gamma} = 50000\mathbf{I}_{m\times m}$ and $\mathbf{K}_\theta = \mathbf{I}_{3\times3}$[3].

[3]Different values in $\mathbf{K}_\theta$ help to tune the estimator's response for deformation coordinates with different scale units (e.g. mixing px with rad).

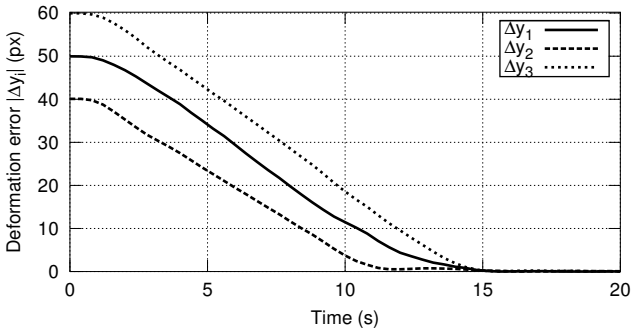


Fig. 5. Magnitude of the point-based deformation errors $\Delta y_i(t)$.

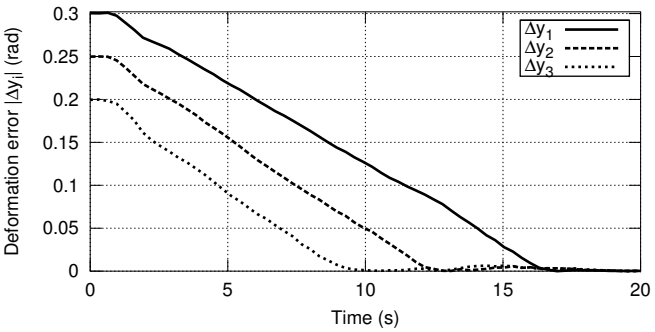


Fig. 6. Magnitude of the angle-based deformation errors $\Delta y_i(t)$.

C. Results

Fig. 4 shows snapshots of the initial and final configurations of two deformation experiments. The magnitude of the error coordinates $\Delta y_i(t)$ for the point-based and angle-based experiments are respectively shown in Fig. 5–6. From these figures we see the effect of our saturated potential control action $\frac{\partial \mathcal{U}}{\partial \mathbf{y}}(\Delta\mathbf{y}(t))$, i.e., an error trajectory that approximates a straight line. The resulting curves of the relative end-effector displacements $\delta\mathbf{x}(t)$ are shown in Fig. 7.

In Fig. 8–9 we present graphical comparisons of the measured deformation flow $\dot{\mathbf{y}}(t)$ and the flow estimated by

$$\mathbf{z}(t) = \widehat{\mathbf{J}}(t)\dot{\mathbf{x}}(t). \tag{32}$$

These figures show that our method closely approximates the *output* deformation flow, and removes noise from the visual measurements.

In the accompanying video, we illustrate the performance of the control method with several uncalibrated 3-DOF deformation tasks.

VI. Conclusions

In this paper, we have proposed a new method to visually servo control the deformation of an unknown elastic object. To avoid identifying the deformation model and camera's parameters, we have first presented a new online algorithm that adaptively estimates the unknown Jacobian matrix of the elastic object. Next, we have derived a dynamic-state feedback velocity control law to smoothly perform the desired deformation.

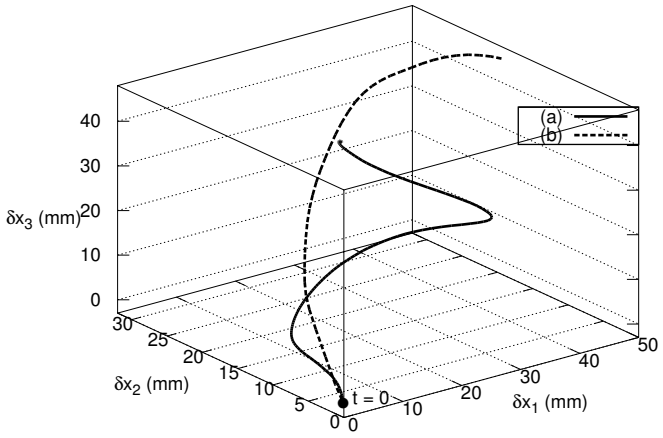


Fig. 7. Three-dimensional trajectory of the relative end-effector displacements $\delta\mathbf{x}(t)$, where we set $\overline{\mathbf{x}} = \mathbf{x}(0)$. The curves (a) and (b) respectively stand for the point-based and angle-based deformation tasks.

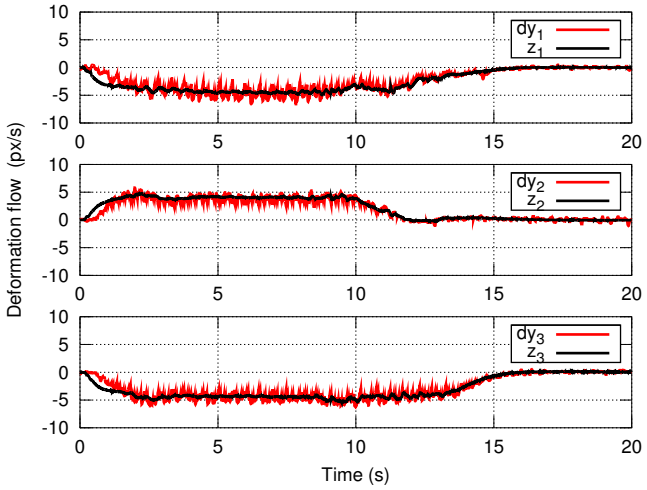


Fig. 8. Comparison of the visually measured deformation flow $\dot{\mathbf{y}}(t)$ and the numerically estimated flow $\mathbf{z}(t)$, of the point-based deformation experiment.

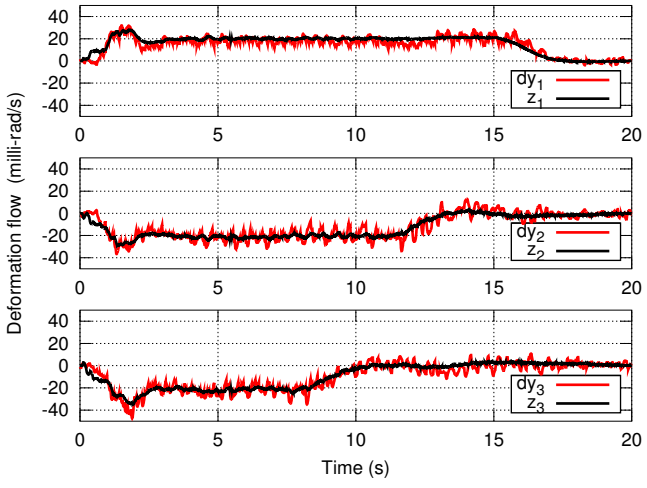


Fig. 9. Comparison of the visually measured deformation flow $\dot{\mathbf{y}}(t)$ and the numerically estimated flow $\mathbf{z}(t)$, of the angle-based deformation experiment.

Slow motion of the manipulator is required by the control method that we have presented. In our experimental study, we have achieved good results with a visual flow of around 4 pixel/second. We must remark that slow motion does not impose severe constraints to many real-life applications, e.g. in surgery. As future research, is the consideration of the nonlinear perspective camera model. We are currently formulating an online estimation algorithm that fully exploits the perspective projection model.

REFERENCES

[1] D. Henrich and H. Wörn, Eds., *Robot manipulation of deformable objects*, ser. Advanced manufacturing. New York: Springer, 2000.

[2] V. Mallapragada, N. Sarkar, and T. Podder, "Toward a robot-assisted breast intervention system," *IEEE/ASME Trans. Mechatronics*, vol. 16, no. 6, pp. 1011–1020, Dec. 2011.

[3] M. Cusumano-Towner, A. Singh, S. Miller, J. O'Brien, and P. Abbeel, "Bringing clothing into desired configurations with limited perception," in *Proc. IEEE Int. Conf. Robotics and Automation*, 2011, pp. 3893–3900.

[4] S. Tokumoto and S. Hirai, "Deformation control of rheological food dough using a forming process model," in *Proc. IEEE Int. Conf. Robotics and Automation*, vol. 2, 2002, pp. 1457–1464.

[5] Z. Wang and S. Hirai, "Modeling and estimation of rheological properties of food products for manufacturing simulations," *J. Food Eng.*, vol. 102, no. 2, pp. 136 – 144, Jan. 2011.

[6] H. Wakamatsu, E. Arai, and S. Hirai, "Knotting/unknotting manipulation of deformable linear objects," *Int. J. Rob. Res.*, vol. 25, no. 4, pp. 371–395, Apr. 2006.

[7] M. Higashimori, K. Yoshimoto, and M. Kaneko, "Active shaping of an unknown rheological object based on deformation decomposition into elasticity and plasticity," in *Proc. IEEE Int. Conf. Robotics and Automation*, 2010, pp. 5120–5126.

[8] M. Shibata and S. Hirai, "Soft object manipulation by simultaneous control of motion and deformation," in *Proc. IEEE Int. Conf. Robotics and Automation*, 2006, pp. 2460–2465.

[9] S. Hirai and T. Wada, "Indirect simultaneous positioning of deformable objects with multi-pinching fingers based on an uncertain model," *Robotica*, vol. 18, no. 1, pp. 3–11, Jan. 2000.

[10] T. Wada, S. Hirai, S. Kawamura, and N. Kamiji, "Robust manipulation of deformable objects by a simple PID feedback," in *Proc. IEEE Int. Conf. Robotics and Automation*, vol. 1, 2001, pp. 85–90.

[11] J. Smolen and A. Patriciu, "Deformation planning for robotic soft tissue manipulation," in *Int. Conf. Advances Computer-Human Interactions*, 2009, pp. 199 –204.

[12] J. Das and N. Sarkar, "Autonomous shape control of a deformable object by multiple manipulators," *J. Intell. & Robot. Syst.*, vol. 62, pp. 3–27, Apr. 2011.

[13] D. Navarro-Alarcon, Y.-H. Liu, J. G. Romero, and P. Li, "Model-free visually servoed deformation control of elastic objects by robot manipulators," *IEEE Trans. Robot., to be published*, 2013.

[14] K. Hosoda and M. Asada, "Versatile visual servoing without knowledge of true Jacobian," in *Proc. IEEE Int. Conf. Intelligent Robots and Systems*, vol. 1, 1994, pp. 186–193.

[15] R. I. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*, 2nd ed. Cambridge University Press, 2004.

[16] S. Hutchinson, G. Hager, and P. Corke, "A tutorial on visual servo control," *IEEE Trans. Robot. Autom.*, vol. 12, no. 5, pp. 651–670, Oct. 1996.

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

[18] Y.-H. Liu, H. Wang, C. Wang, and K. K. Lam, "Uncalibrated visual servoing of robots using a depth-independent interaction matrix," *IEEE Trans. Robot.*, vol. 22, no. 4, pp. 804–817, Aug. 2006.

[19] J.-J. Slotine and W. Li, *Applied Nonlinear Control*. New Jersey, USA: Prentice Hall, 1991.

[20] J. E. Marsden and T. Ratiu, *Introduction to Mechanics and Symmetry*. USA: Texts in Applied Mathematics, Springer-Verlag, 1994.