Shape Servoing of a Soft Object Using Fourier Series and a Physics-based Model

Fouad Makiyeh, François Chaumette, Maud Marchal, Alexandre Krupa

Shape Servoing of a Soft Object Using Fourier Series and a Physics-based Model

Fouad Makiyeh[1], François Chaumette[1], Maud Marchal[2], Alexandre Krupa[1]

***Abstract*— In this paper, we propose a physics-based robot controller to deform a soft object toward a desired 3D shape using a limited number of handling points. For this purpose, the shape of the deformable object is represented using Fourier descriptors. We derive the analytical relation that provides the variation of the Fourier coefficients as a function of the movements of the handling points by considering a mass-spring model (MSM). A control law is then designed from this relation. Since the MSM provides an approximation of the object behavior, which in practice can lead to a drift between the object and its model, an online realignment of the model with the real object is performed by tracking its surface from data provided by a remote RGB-D camera. Simulation results validate the approach for the case where many points interact on a 2D soft object while experimental results obtained with two robotic arms demonstrate the autonomous shaping of a 3D soft object.**

I. INTRODUCTION

While traditional robotic applications focus on manipulating rigid objects, interacting with soft objects remains a more challenging task due to their deformable behavior. Robotic manipulation of soft objects has already been considered in applications such as robotic surgical assistance, clothing, and food handling, as well as in the automotive industry, to name a few.

Recently, in [1], we proposed a vision-based and model-based control strategy to indirectly position one point belonging to a soft object, referred as the feature point, by acting on a distant point, referred as the manipulated point. This approach is useful when one point needs to be automatically positioned. However, when the entire shape of the object needs to be controlled, it is no longer appropriate as it would require a large number of feature points for representing the object shape. To address this issue, a global representation of the object shape can be used. For example, Collewet et al. [2] used Fourier coefficients as features in image-based visual servoing for describing the contour of a rigid object. Chaumette [3] described the object contours from image moments and Yazicioglu et al. [4] approximated the object boundaries with polynomials, to name a few. While these approaches have been used in visual servoing to control the displacement of a camera with respect to a rigid object, Navarro et al. used truncated Fourier series to approximate the 2D contour of a soft object and to control its 2D shape in the image plane [5]. Recently, Qi et al. [6] used a compact feedback vector built from 2D contour moments and applied sliding mode control to servo the shape of a 3D deformable object. One limitation of these two previous approaches is that they only control the boundary contour observed from a particular incidence view of the camera. Indeed, for a 3D object, there exists an infinite number of possible outer boundary 2D contours since they depend on the relative orientation between the camera and the object. Our objective in this work is therefore to develop a new approach for controlling the shape of the complete visible 3D surface of the object and not only its outer 2D contour. Our goal is to align the 3D surface shape of the object to a desired 3D surface shape by controlling the motion of the robotic manipulators deforming it.

To perform this task, two key components have to be examined. The first one is the determination of the relation that links the variation of the extracted features to the displacement of the 3D points belonging to the soft object. This depends on the definition of the shape parametrization used. The second one involves determining the relation between the displacement of any 3D point belonging to the soft object and the motions applied to the available manipulated points. Some works have already focused on these topics and they can be classified in two categories. The first one concerns the model-free approaches, which do not require neither prior information on deformation behavior, nor physics-based models. For instance, Navarro-Alarcon et al. developed a series of works for controlling the configuration of deformable objects using visual servoing. In these works, the shape of the object is described by a set of visual features extracted from a 2D image corresponding, for example, to multiple 2D points, the relative angle of a line of interest constructed from two 2D points, the curvature of a contour of interest defined from three 2D points [7], [8], or the object 2D contour [9]. The shape servoing task is performed by numerically estimating a deformation Jacobian matrix, which relates the visual feature variations to the motions of the manipulated points, using data provided by a RGB camera [7]–[9]. These works present important advantages since they do not rely on any deformation model of the object. However, they are sensitive to measurement noise since the estimation of the deformation Jacobian is only based on these measurements. The second category of approaches concerns the physics-based methods, where a model approximating the physical behavior of the object

*This work was supported by the GentleMAN (299757) and the BIFROST (313870) projects funded by The Research Council of Norway. Experiments presented in this paper were carried out thanks to a platform of the Robotex 2.0 French research infrastructure.

[1]F. Makiyeh, A. Krupa, F. Chaumette are with Inria, Univ. Rennes, CNRS, IRISA, Campus de Beaulieu, 35042 Rennes, France. Firstname.Name@inria.fr

[2]M. Marchal is with Univ. Rennes, INSA, IRISA, Inria, CNRS, France and IUF. Maud.Marchal@irisa.fr

is considered. Recently, Shetab-Bushehri et al. [10] used the As-Rigid-As-Possible deformation model to approximate the object shape and they considered 3D points belonging to the object as feedback for the shape servoing task. Other types of physics-based models such as the finite element model (FEM) [11] and the mass-spring model (MSM) [12], [13] were also used. Das and Sarkar [12] developed a method for handling a 2D deformable object relying on MSM with multiple robotic manipulators. The shape of the object is controlled by applying an optimization technique over the model boundary points, but only simulation results were presented. Das et al. [13] used three robotic fingers to deform an object represented by a MSM for driving a single internal point to a desired position. They used a proportional integrator (PI) controller for each finger and only simulation results were presented. Other approaches used FEM instead of MSM, such as Ficuciello et al. [11] who presented an approach for the dexterous manipulation of 3D deformable objects in an open-loop system.

In this work, we present a new approach that controls multiple manipulator points to drive the entire shape of a soft object parametrized by a set of Fourier coefficients to a desired configuration. We previously proposed a physics-based control law to move a single 3D feature point to a desired location by acting on a different manipulated point [1]. In this study, we expand our method to simultaneously control the position of several 3D feature points by acting on multiple manipulation points. Additionally, we employ Fourier series to describe the surface of a 3D object using spherical coordinates. In case of planar objects, Fourier series can also be used from the polar coordinates of the 2D contour shape. We use the MSM to simulate the object physical behavior and utilize a RGB-D camera to online track its current shape and correct any possible drift between the simulated model and the actual object. The main contribution of this work is the analytic derivation of the relation that links the variation of the Fourier coefficients, which represent the object shape parameters, to the movements of multiple manipulated points. Based on this relation, a closed-loop control law is then developed to automatically deform the soft object toward a desired shape.

An overview of our approach is presented in Fig. 1. The main steps can be outlined as:

1) Compute the velocities to be applied to the manipulated points (see Section II-E).
2) Update the physics-based model using the internal and external forces acting on it (see Section II-A).
3) Reduce the drift between the model and the real object by imposing external constraints on the model surface from data provided by the RGB-D camera (see Section IV-A).
4) Compute the error between the feature vector representing the object surface and the one corresponding to the desired shape (see Sections II-C and II-E). If this error is not low enough, go to step 1).

The paper is organized as follows: Section II presents briefly the MSM physics-based model, details the modelling of the features representing the object shape, and develops the control law. Section III and Section IV show simulation and experimental results respectively. Finally, Section V ends the paper with a conclusion on the proposed approach and introduces future works.

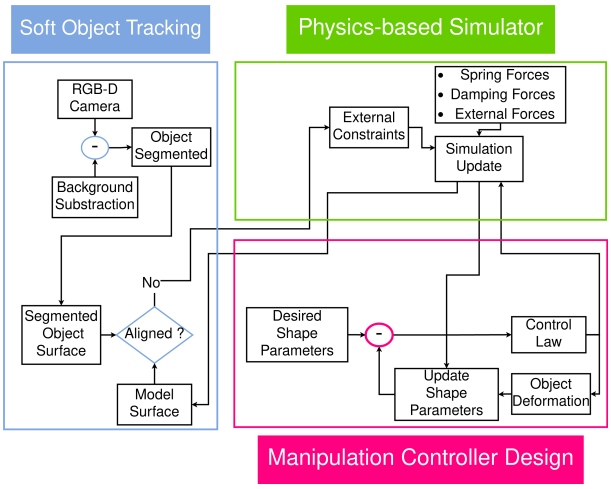


Fig. 1. Block diagram of the approach: visual tracking part in blue, physics-based simulator in green, and closed-loop control scheme in red.

II. Manipulation Controller Design

A. MSM model

This part presents a brief description of the mass-spring model used to approximate the soft object behavior. It has been chosen since it is simple to implement and it has already been applied to simulate and deform soft objects as in [1]. The position of any point P_i, where $i \in \mathcal{N} = [1, ..., N]$ and N is the number of model points, with corresponding mass m_i, coordinate $\boldsymbol{x}_i$ and velocity $\dot{\boldsymbol{x}}_i$, is given at time $t = t + dt$ by:

$$\boldsymbol{x}_i(t+dt) = \boldsymbol{x}_i(t) + \frac{dt^2}{m_i}\boldsymbol{f_s}_i(t) + (dt - \frac{dt^2}{m_i}D_v)\dot{\boldsymbol{x}}_i(t) + \frac{dt^2}{m_i}\boldsymbol{f_e}_i(t) \quad (1)$$

with dt the simulation step time and $\boldsymbol{f_e}_i = \boldsymbol{f_G}_i + \boldsymbol{f_g}_i + \boldsymbol{f_c}_i$. $\boldsymbol{f_G}_i$ and $\boldsymbol{f_g}_i$ are respectively the 3D gravitational force and the normal force exerted by the support on which the object lies, while $\boldsymbol{f_c}_i$ is a 3D force vector representing additional external constraints. This force is applied each time a drift between the model and the real object is detected (see Section IV-A). Finally, $\boldsymbol{f_s}_i$ is the 3D force vector acting on P_i due to springs with stiffness $\boldsymbol{K}_{ij}$ connecting P_i to its neighbors, P_j, $\forall j \in \boldsymbol{\nu}_i \subset \mathcal{N}$, given by:

$$\boldsymbol{f_s}_i = \sum_{j \in \boldsymbol{\nu}_i} \boldsymbol{f_s}_{ij} = \sum_{j \in \boldsymbol{\nu}_i} \Gamma_{ij}\boldsymbol{K}_{ij}(\boldsymbol{x}_j - \boldsymbol{x}_i) \quad (2)$$

where $\Gamma_{ij} = \frac{\|\boldsymbol{x}_i - \boldsymbol{x}_j\| - l^0_{ij}}{\|\boldsymbol{x}_j - \boldsymbol{x}_i\|}$ with l^0_{ij} the rest length of the spring connecting P_i and P_j, and $\boldsymbol{\nu}_i$ denotes the indices in $\mathcal{N}$ of the points in the neighborhood of P_i.

B. Problem formulation

The main objective of this work is to deform a 3D soft object with w surface points, represented by $\boldsymbol{x}_o = (\boldsymbol{x}_{o_1}, ..., \boldsymbol{x}_{o_w}); \forall 1 \leq i \leq w; o_i \in \mathcal{N}$, where o_i is the index of each surface point in $\mathcal{N}$. Let $\boldsymbol{s}$ be a feature vector representing the 3D surface $\boldsymbol{x}_o$ of the object. The goal is to drive $\boldsymbol{s}$ to a desired target shape $\boldsymbol{s}^*$ by simultaneously manipulating M points on the object, represented by $[P_{m_1}, ..., P_{m_M}]$, with coordinates $(\boldsymbol{x}_{m_1}, ..., \boldsymbol{x}_{m_M})$. By determining how $\boldsymbol{s}$ changes due to the motion of these manipulated points, the required motions to be applied on them for achieving the desired shape can be determined as a classical visual servoing problem [14].

C. Surface shape parameters

By expressing the points $\boldsymbol{x}_{o_l}, l \in [1, ..., w]$ on the object surface with spherical coordinates, each component can be written as:

$$\begin{cases} x_{o_l} &= x_o^g + \rho(\theta_{o_l}, \phi_{o_l}) \cos(\phi_{o_l}) \sin(\theta_{o_l}) \\ y_{o_l} &= y_o^g + \rho(\theta_{o_l}, \phi_{o_l}) \sin(\phi_{o_l}) \sin(\theta_{o_l}) \\ z_{o_l} &= z_o^g + \rho(\theta_{o_l}, \phi_{o_l}) \cos(\theta_{o_l}) \end{cases} \tag{3}$$

with $\boldsymbol{x}_o^g = (x_o^g, y_o^g, z_o^g)$ the 3D coordinates of the centroid of $\boldsymbol{x}_o$, $\rho(\theta_{o_l}, \phi_{o_l})$ the radial distance $\|\boldsymbol{x}_{o_l} - \boldsymbol{x}_o^g\|$ of the point P_{o_l} from $\boldsymbol{x}_o^g$, $\theta_{o_l} = \arccos(z_{o_l}/\rho(\theta_{o_l}, \phi_{o_l}))$ the polar angle and $\phi_{o_l} = \text{atan2}(y_{o_l}, x_{o_l})$ the azimuthal angle. $\rho(\theta, \phi)$ is a closed continuous periodic function, which can be represented as a sum of cosine and sine functions with increasing frequencies, known as Fourier descriptors. It can be approximated using Fourier series as following:

$$\begin{aligned} \rho(\theta_{o_l}, \phi_{o_l}) = \sum_{i=0}^{p} \sum_{j=0}^{q} [&a_{ij} c(i\theta_{o_l}) c(j\phi_{o_l}) + b_{ij} c(i\theta_{o_l}) s(j\phi_{o_l}) \\ &+ c_{ij} s(i\theta_{o_l}) c(j\phi_{o_l}) + d_{ij} s(i\theta_{o_l}) s(j\phi_{o_l})] \end{aligned} \tag{4}$$

where c stands for cos, s for sin, a_{ij}, b_{ij}, c_{ij} and d_{ij} are the Fourier coefficients, and p and q are the number of harmonics corresponding to θ and ϕ respectively. Hence, $\boldsymbol{s}$ representing the shape parameters is selected as:

$$\boldsymbol{s} = (x_o^g, y_o^g, z_o^g, a_{00}, ..., a_{pq}, b_{01}, ..., b_{pq}, c_{pq}, d_{pq}) \tag{5}$$

As noticed from (5), the dimension k of $\boldsymbol{s}$, ($k = 4(p+1)(q+1) - 2(p+q)$) is significantly less than the number of object surface points for low values of p and q.

Regarding the components of $\boldsymbol{s}$, they are obtained as follows:

1) The centroid of the model surface points is first computed: $\boldsymbol{x}_o^g = \frac{1}{w} \sum_{i=1}^{w} \boldsymbol{x}_{o_i}$
2) The model surface points are centered with respect to the centroid: $\hat{\boldsymbol{x}}_o = \boldsymbol{x}_o - \boldsymbol{x}_o^g$
3) The Fourier coefficients are then computed by a least squares method. In theory, only $(dim(\boldsymbol{s}) - 3)$ surface points are needed, but to increase robustness to measurement noise, all w points are considered in the computation.

Once $\boldsymbol{s}$ has been calculated, an approximation of the object surface can be reconstructed by setting the range of θ to [0, π] and ϕ to [0, 2π].

D. Modeling

By approximating the physical behavior of the object using the MSM, we demonstrated in [1] that the velocity $\dot{\boldsymbol{x}}_f$ of a feature point P_f due to the velocity $\dot{\boldsymbol{x}}_m$ applied on a manipulated point P_m is given by:

$$\dot{\boldsymbol{x}}_f = \boldsymbol{\gamma}_{fm} \dot{\boldsymbol{x}}_m + \boldsymbol{\delta}_f \tag{6}$$

where the vector $\boldsymbol{\delta}_f = \boldsymbol{r}_f / \Delta t$ represents the transient dynamics of the system and Δt denotes the step time of the control law. The 3×3 matrix $\boldsymbol{\gamma}_{fm}$ is the velocity ratio between $\dot{\boldsymbol{x}}_f$ and $\dot{\boldsymbol{x}}_m$ and is given at each time t by [1]:

$$\begin{aligned} \boldsymbol{\gamma}_{fm}(t + dt) = \boldsymbol{\gamma}_{fm}(t) &+ \frac{dt^2}{m_f} \boldsymbol{\gamma}_{\boldsymbol{s}f}(t) \\ &+ (1 - \frac{dt}{m_f} D_v)(\boldsymbol{\gamma}_{fm}(t) - \boldsymbol{\gamma}_{fm}(t - dt)) \end{aligned} \tag{7}$$

with

$$\boldsymbol{\gamma}_{\boldsymbol{s}f}(t) = \sum_{F \in \boldsymbol{\nu}_f} \frac{\partial \boldsymbol{f}_{\boldsymbol{s}fF}}{\partial \boldsymbol{x}_F} (\boldsymbol{\gamma}_F(t) - \boldsymbol{\gamma}_f(t))$$

where $\boldsymbol{\nu}_f$ is the set of points in the neighborhood of P_f and $\boldsymbol{f}_{\boldsymbol{s}fF}$ is the spring forces between P_f and its neighbors. Finally, the form of $\boldsymbol{r}_f$ is quite complex but fully given in [1]. Let us note that $\boldsymbol{r}_f = \boldsymbol{0}$ at the beginning and at the end of the servo.

Eq. (6) is valid when a single manipulated point is considered. When several manipulated points deform the object simultaneously, the velocity of a feature point P_f is obtained from the combination of the motions of these manipulated points. More precisely, when M manipulators $(P_{m_1}, P_{m_2}, ..., P_{m_M})$ are deforming the object simultaneously with corresponding velocities $\dot{\boldsymbol{x}}_m = (\dot{\boldsymbol{x}}_{m_1}, ..., \dot{\boldsymbol{x}}_{m_M})$, we can show by similarity with (6) that the velocity of P_f is given by:

$$\dot{\boldsymbol{x}}_f = \sum_{l=1}^{M} \boldsymbol{\gamma}_{f\,m_l} \dot{\boldsymbol{x}}_{m_l} + \boldsymbol{\delta}_f \tag{8}$$

where $\dot{\boldsymbol{x}}_{m_l}$ is the applied velocity on P_{m_l}, $l \in [1, ..., M]$. $\boldsymbol{\gamma}_{f\,m_l}$ is calculated in the same way as presented in (7). Additionally, $\boldsymbol{\delta}_f$ can also be computed similarly as in [1].

From (8), we directly obtain the velocity of all points belonging to the model surface:

$$\dot{\boldsymbol{x}}_o = \boldsymbol{L_o}\, \dot{\boldsymbol{x}}_m + \boldsymbol{\delta_o} \tag{9}$$

with

$$\boldsymbol{L_o} = \begin{bmatrix} \boldsymbol{\gamma}_{o_1 m_1} & \cdots & \boldsymbol{\gamma}_{o_1 m_M} \\ \boldsymbol{\gamma}_{o_2 m_1} & \cdots & \boldsymbol{\gamma}_{o_2 m_M} \\ \vdots & \ddots & \vdots \\ \boldsymbol{\gamma}_{o_w m_1} & \cdots & \boldsymbol{\gamma}_{o_w m_M} \end{bmatrix}, \boldsymbol{\delta_o} = \begin{bmatrix} \boldsymbol{\delta}_{o_1} \\ \vdots \\ \boldsymbol{\delta}_{o_w} \end{bmatrix}, \dot{\boldsymbol{x}}_o = \begin{bmatrix} \dot{\boldsymbol{x}}_{o_1} \\ \vdots \\ \dot{\boldsymbol{x}}_{o_w} \end{bmatrix}.$$

The ultimate goal is to find the variation $\dot{\boldsymbol{s}}$ of the features with respect to the motions $\dot{\boldsymbol{x}}_m$ of the manipulated points, through the following form:

$$\dot{\boldsymbol{s}} = \boldsymbol{L_s}\, \dot{\boldsymbol{x}}_m + \boldsymbol{\delta_s} \tag{10}$$

By deriving (3) and (4), we obtain the motion $\dot{\boldsymbol{x}}_{o_l} = (\dot{x}_{o_l}, \dot{y}_{o_l}, \dot{z}_{o_l})$ of each point belonging to the object surface, as a function of $\dot{\boldsymbol{s}}$, $\dot{\theta}_{o_l}$ and $\dot{\phi}_{o_l}$. However, we have to eliminate $\dot{\theta}$ and $\dot{\phi}$ in order to express the motions of these points solely in terms of $\dot{\boldsymbol{s}}$. For that, by summing the three equations in (3) after introducing terms α_{o_l}, β_{o_l} and σ_{o_l}, we obtain:

$$\alpha_{o_l}\dot{x}_{o_l} + \beta_{o_l}\dot{y}_{o_l} + \sigma_{o_l}\dot{z}_{o_l} = [\alpha_{o_l}\ \beta_{o_l}\ \sigma_{o_l}\ \nu_{o_l}\frac{\partial\rho(\theta_{o_l},\phi_{o_l})}{\partial \boldsymbol{q}}]\dot{\boldsymbol{s}} \tag{11}$$

$\forall l \in [1, ..., w]$ with

$$\begin{aligned}
&\alpha_{o_l} = b_1c_2 - b_2c_1\\
&\beta_{o_l} = a_2c_1 - a_1c_2\\
&\sigma_{o_l} = a_1b_2 - a_2b_1\\
&\nu_{o_l} = \alpha_{o_l}[(\cos(\phi_{o_l}) + \beta_{o_l}\sin(\phi_{o_l}))\sin(\theta_{o_l}) + \sigma_{o_l}\cos(\theta_{o_l})]\\
&\boldsymbol{q} = (a_{00}, ..., a_{pq}, b_{01}, ..., b_{pq}, c_{pq}, d_{pq})
\end{aligned}$$

and

$$\begin{aligned}
a_1 &= (\frac{\partial\rho(\theta_{o_l},\phi_{o_l})}{\partial\theta_{o_l}}\sin(\theta_{o_l}) + \rho(\theta_{o_l},\phi_{o_l})\cos(\theta_{o_l}))\cos(\phi_{o_l}),\\
b_1 &= (\frac{\partial\rho(\theta_{o_l},\phi_{o_l})}{\partial\theta_{o_l}}\sin(\theta_{o_l}) + \rho(\theta_{o_l},\phi_{o_l})\cos(\theta_{o_l}))\sin(\phi_{o_l}),\\
c_1 &= \frac{\partial\rho(\theta_{o_l},\phi_{o_l})}{\partial\theta_{o_l}}\cos(\theta_{o_l}) - \rho(\theta_{o_l},\phi_{o_l})\sin(\theta_{o_l}),\\
a_2 &= (\frac{\partial\rho(\theta_{o_l},\phi_{o_l})}{\partial\phi_{o_l}}\cos(\phi_{o_l}) - \rho(\theta_{o_l},\phi_{o_l})\sin(\phi_{o_l}))\sin(\theta_{o_l}),\\
b_2 &= (\frac{\partial\rho(\theta_{o_l},\phi_{o_l})}{\partial\phi_{o_l}}\sin(\phi_{o_l}) + \rho(\theta_{o_l},\phi_{o_l})\cos(\phi_{o_l}))\sin(\theta_{o_l}),\\
c_2 &= \frac{\partial\rho(\theta_{o_l},\phi_{o_l})}{\partial\phi_{o_l}}\cos(\theta_{o_l}).
\end{aligned}$$

By processing similarly with (9) and identifying the equation obtained with (11), we get:

$$\begin{aligned}
\begin{bmatrix} \alpha_{o_1}\dot{x}_{o_1} + \beta_{o_1}\dot{y}_{o_1} + \sigma_{o_1}\dot{z}_{o_1} \\ \vdots \\ \alpha_{o_w}\dot{x}_{o_w} + \beta_{o_w}\dot{y}_{o_w} + \sigma_{o_w}\dot{z}_{o_w} \end{bmatrix} &= \boldsymbol{A_o}\,\dot{\boldsymbol{x}}_m + \boldsymbol{b_o}\\
&= \boldsymbol{C_o}\,\dot{\boldsymbol{s}}
\end{aligned} \tag{12}$$

with

$$\boldsymbol{A_o} = \begin{bmatrix} & & & & & \\ [\alpha_{o_1} & \beta_{o_1} & \sigma_{o_1}] & [\boldsymbol{\gamma}_{o_1 m_1} & \cdots & \boldsymbol{\gamma}_{o_1 m_M}] \\ & \vdots & & & \vdots & \\ \underbrace{[\alpha_{o_w} \quad \beta_{o_w} \quad \sigma_{o_w}]}_{1\text{x}3} & & & \underbrace{[\boldsymbol{\gamma}_{o_w m_1} & \cdots & \boldsymbol{\gamma}_{o_w m_M}]}_{3\text{x}3M} \end{bmatrix}$$

$$\boldsymbol{b_o} = \begin{bmatrix} [\alpha_{o_1} & \beta_{o_1} & \sigma_{o_1}]\,\boldsymbol{\delta}_{o_1} \\ & \vdots & \\ [\alpha_{o_w} & \beta_{o_w} & \sigma_{o_w}]\,\boldsymbol{\delta}_{o_w} \end{bmatrix}$$

$$\boldsymbol{C_o} = \begin{bmatrix} \alpha_{o_1} & \beta_{o_1} & \sigma_{o_1} & \nu_{o_1}\frac{\partial\rho(\theta_{o_1},\phi_{o_1})}{\partial\boldsymbol{q}} \\ \vdots & \vdots & \ddots & \vdots \\ \alpha_{o_w} & \beta_{o_w} & \sigma_{o_w} & \nu_{o_w}\frac{\partial\rho(\theta_{o_w},\phi_{o_w})}{\partial\boldsymbol{q}} \end{bmatrix}$$

By identifying (12) with (10), we finally obtain:

$$\boldsymbol{L_s} = \boldsymbol{C_o}^{+}\boldsymbol{A_o} \text{ and } \boldsymbol{\delta_s} = \boldsymbol{C_o}^{+}\boldsymbol{b_o} \tag{13}$$

with $\boldsymbol{C_o}^{+}$ the Moore-Penrose pseudo-inverse of $\boldsymbol{C_o}$. Matrix $\boldsymbol{C_o}$ is of dimension $w \times k$ and is full rank since the number w of surface points is highly larger than the number of harmonics. $\boldsymbol{L_s}$ can be considered as the interaction matrix by similarity with the classical visual servoing framework [14], while $\boldsymbol{\delta_s}$ is a feed-forward term that does not appear in classical shape servoing works.

E. Control scheme

This section presents the control law to be applied to the manipulated points in order to drive the object to its desired shape. Similarly to what is classically done in visual servoing, by specifying a decoupled exponential decay for the visual features $\boldsymbol{s}$ to their desired values $\boldsymbol{s}^*$, that is, $\dot{\boldsymbol{s}} = -\lambda\ (\boldsymbol{s} - \boldsymbol{s}^*)$ where λ is a positive gain, we obtain from (10):

$$\dot{\boldsymbol{x}}_m = -\lambda\boldsymbol{L_s}^{+}\ (\boldsymbol{s} - \boldsymbol{s}^*) - \boldsymbol{L_s}^{+}\boldsymbol{\delta_s} \tag{14}$$

Regarding the achievement of the deformation task, it depends on the number of harmonics p and q. When $k > 3M$, the controller (14) is under-actuated and cannot guarantee the asymptotic convergence of the system. Conversely, when $k \leq 3M$, the controller is either fully-actuated or over-actuated and therefore stability and convergence of $\boldsymbol{s}$ towards $\boldsymbol{s}^*$ is theoretically guaranteed. Let us note however that this does not necessarily mean that the full object will reach its desired shape, but only its representation by the selected Fourier coefficients. So, the choice of number of harmonics to be injected in $\boldsymbol{s}$ depends on the complexity of the shape to be obtained, while complex deformations require more harmonics and consequently more manipulated points.

F. Specific application to planar objects

The method described previously can easily be applied to planar objects. The shape of a 2D object is represented by its contour $\boldsymbol{x}_c$ containing w_c contour points $\boldsymbol{x}_{c_l}$. It can be parameterized by its centroid (x_c^g, y_c^g) and radial distance $\rho(\theta)$ that now only depends on a single angle θ. Doing so, any point $\boldsymbol{x}_{c_l}$ in $\boldsymbol{x}_c$ can be expressed using polar coordinates as follows:

$$\boldsymbol{x}_{c_l} = (x_c^g + \rho(\theta_{c_l})\cos(\theta_{c_l}), y_c^g + \rho(\theta_{c_l})\sin(\theta_{c_l})). \tag{15}$$

As $\rho(\theta)$ is periodic, it can be approximated using Fourier series as following:

$$\rho(\theta_{c_l}) = a_0^c + \sum_{i=1}^{p_c}[a_i^c\cos(i\theta_{c_l}) + b_i^c\sin(i\theta_{c_l})] \tag{16}$$

with p_c the number of harmonics corresponding to θ. The features $\boldsymbol{s}_c$ are thus naturally selected as:

$$\boldsymbol{s}_c = (x_c^g, y_c^g, a_0^c, a_1^c, b_1^c, ..., a_{p_c}^c, b_{p_c}^c) \tag{17}$$

Following the same reasoning as in Section II-D, the variation of $\boldsymbol{s}_c$ with respect to the motions of the manipulated points $\dot{\boldsymbol{x}}_m$ can be expressed as in (10):

$$\dot{\boldsymbol{s}_c} = \boldsymbol{L_{s_c}}\, \dot{\boldsymbol{x}}_m + \boldsymbol{\delta_{s_c}} \tag{18}$$

with $\boldsymbol{L_{s_c}} = \boldsymbol{C_c}^{+}\boldsymbol{A_c}$ and $\boldsymbol{\delta_{s_c}} = \boldsymbol{C_c}^{+}\boldsymbol{b_c}$ where

$$\boldsymbol{A_c} = \begin{bmatrix} [\alpha_{c_1} & \beta_{c_1}][\boldsymbol{\gamma}_{c_1 m_1} & \cdots & \boldsymbol{\gamma}_{c_1 m_M}] \\ & & \vdots & \\ [\alpha_{c_{w_c}} & \beta_{c_{w_c}}][\boldsymbol{\gamma}_{c_{w_c} m_1} & \cdots & \boldsymbol{\gamma}_{c_{w_c} m_M}] \end{bmatrix},$$

$$\boldsymbol{b_c} = \begin{bmatrix} [\alpha_{c_1} & \beta_{c_1}]\, \boldsymbol{\delta}_{c_1} \\ & \vdots \\ [\alpha_{c_{w_c}} & \beta_{c_{w_c}}]\, \boldsymbol{\delta}_{c_{w_c}} \end{bmatrix},$$

$$\boldsymbol{C_c} = \begin{bmatrix} \alpha_{c_1} & \beta_{c_1} & \nu_{c_1} \frac{\partial \rho(\theta_{c_1})}{\partial \boldsymbol{q}_c} \\ \vdots & \vdots & \vdots \\ \alpha_{c_{w_c}} & \beta_{c_{w_c}} & \nu_{c_{w_c}} \frac{\partial \rho(\theta_{c_{w_c}})}{\partial \boldsymbol{q}_c} \end{bmatrix},$$

and

$$\begin{aligned} \alpha_{c_l} &= \rho(\theta_{c_l})\cos(\theta_{c_l}) + \frac{\partial \rho(\theta_{c_l})}{\partial \theta_{c_l}} \sin(\theta_{c_l}), \\ \beta_{c_l} &= \rho(\theta_{c_l})\sin(\theta_{c_l}) - \frac{\partial \rho(\theta_{c_l})}{\partial \theta_{c_l}} \cos(\theta_{c_l}), \\ \nu_{c_l} &= \alpha_{c_l}(cos(\theta_{c_l}) + \beta_{c_l} sin(\theta_{c_l})), \\ \boldsymbol{q}_c &= (a_0^c, a_1^c, b_1^c, ..., a_{p_c}^c, b_{p_c}^c). \end{aligned}$$

The closed-loop control law to be applied by the robotic manipulators is directly deduced from (18):

$$\dot{\boldsymbol{x}}_m = -\lambda \boldsymbol{L}_{\boldsymbol{s_c}}^{+}\, (\boldsymbol{s}_c - \boldsymbol{s}_c{}^{*}) - \boldsymbol{L}_{\boldsymbol{s_c}}^{+} \boldsymbol{\delta}_{\boldsymbol{s_c}}, \tag{19}$$

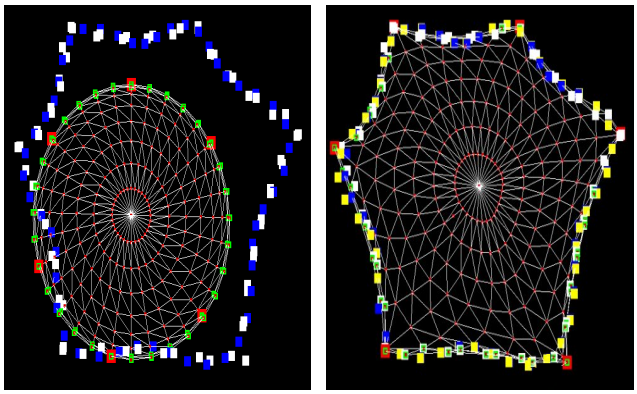
III. Simulation results

In this section, the proposed controller is tested in simulation for deforming a 2D object to a desired shape using Fourier descriptors.

As already mentioned, the number of selected harmonics influences the accuracy of the deformation task. In other terms, it is crucial to choose an appropriate number of harmonics for obtaining a correct approximation of an object contour. For complex shapes, selecting a higher number of harmonics will improve the accuracy of the approximation. However, when the contour can be approximated using a low number of harmonics all along its deformation, there is no need to select a large number, since it will make the system more and more under-actuated without providing any significant information. Additionally, the desired shape must be feasible. This can be obtained through simulation by applying arbitrary motions on the chosen manipulated points in order to deform the object. The resulting contour can then be selected as the desired one since we are sure that it is feasible.

The chosen 2D object is depicted in Fig. 2. It is represented using a MSM with 256 points, where the small red points are the model points while the 6 larger red points denote the manipulated points that have been uniformly distributed around the object. Green points denote contour points ($w_c = 32$) and the white lines correspond to the springs between the model points. The white points represent the target points, which correspond to the green points acquired after manually deforming the object to obtain a feasible desired shape. The blue ones depict the desired reconstructed contour using $p_c = 8$ harmonics. The yellow points on Fig. 2.b correspond to the reconstructed contour at the end of the servo using the same number of 8 harmonics. We can notice that the yellow points fit with the green and blue ones and the green points align with the white points, showing the success and the accuracy of the system using 8 harmonics.

The physical parameters of the object have been chosen as follows. The mass of each point is $m_i = 0.35g$. The stiffness matrix $\boldsymbol{K}_{ij}$ depends on two constant values: $\boldsymbol{K}_{ij} = diag(k_x, k_y)$ with $k_x = k_y = 13N/m$. The damping has been chosen according to [1] as $D_v = 2\sqrt{m_i k_x} = 0.13Ns/m$. Finally, we have selected $\lambda = 0.9$ as gain for the control scheme.



(a) (b)

Fig. 2. Deformation of a planar object: (a) initial and desired configurations, (b) final configuration (see text for explanations on color code).

Different simulations have been performed by selecting a different number of harmonics ($p_c = 6, 8$ and 10), while the number of manipulated points is always the same ($M = 6$). The purpose of testing these different numbers of harmonics is to study their effects on the deformation performance. The greater the number of harmonics, the better the reconstruction of the contour. However, from a certain amount of p_c, the reconstruction is perfect and we no longer need to increase it. To correctly reconstruct the contour of the desired shape shown in Fig. 2, at least 6 harmonics are needed. For that, we test $p_c = 6$. Additionally, to achieve a more accurate approximation, we also test two larger numbers of harmonics, $p_c = 8$ and 10. Moreover, we also aim to evaluate the achievement of the task when the system is under-actuated (here, we have 12 DOF while $k_c = dim(\boldsymbol{s}_c) = 2p_c + 3$).

The time evolution of the error norm $\|\boldsymbol{s}_c - \boldsymbol{s}_c{}^{*}\|$ for the different number of harmonics is presented in Fig. 3. The error norm converges exponentially to nearly zero when $p_c = 6$, despite the system being under-actuated (green plot). For the other cases (blue and black plots), a very small residual

remains at the convergence of the system.

Furthermore, the difference between the actual and desired contour is shown in Fig. 4 for the different values of p_c. This difference has been computed using the Hausdorff distance [15] by considering all contour points. We can notice a nice exponential decrease of this error in all cases. After convergence, the remaining error is less than 3% of the initial error using 6 harmonics (green plot) while it is reduced to less than 2% for $p_c = 8$ and 10 (blue and black plots). Since the residual of the Hausdorff distance is almost the same for these two harmonics, using 8 harmonics is here sufficient to achieve the task with high accuracy.

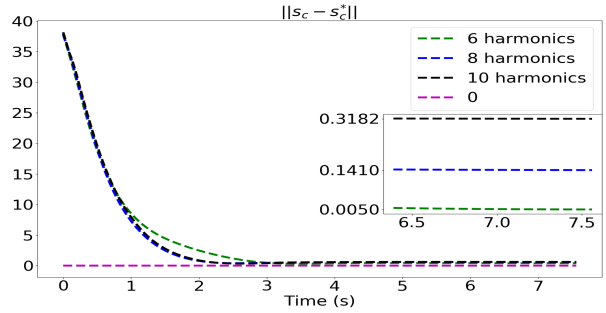


Fig. 3. Evolution of the error norm for different numbers of harmonics and using 6 manipulated points.

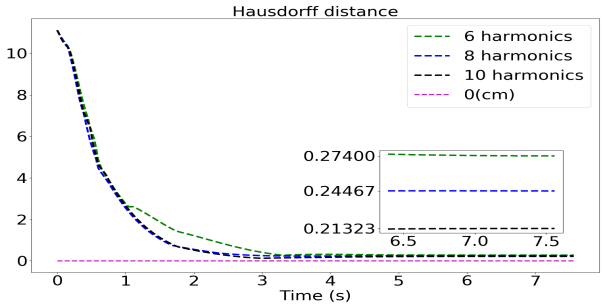


Fig. 4. Evolution of the Hausdorff distance between the actual and desired contours for different numbers of harmonics and using 6 manipulated points.

IV. Real Experiments

A. Setup & Implementation

Our experimental setup consists of two 6-DOF robotic arms from ADEPT ($M = 2$), a Viper 850 and a Viper 650. Two rigid sticks are attached to the robots, used as tools for the deformation process. These sticks are in contact with the deformable object and the contact points are used as the manipulated points of the soft object. The visual perception of the object is performed with a remote RGB-D Intel Realsense-D435 camera. It acquires around 3.10^5 3D points at 30 frames per second (fps).

A simulator has been developed in C++ to simulate the deformable object model and update its point positions at a frequency of 600 Hz, *i.e.,* $dt = 1.66\ ms$. The velocities computed by the control scheme (14) are initially expressed in the camera frame since the data are measured in this frame. Thanks to a classical hand-eye calibration (the camera is fixed in the workspace), these velocities are then expressed in their respective end-effector frame and applied to the manipulated points. The control law is executed at the same frequency as the camera, which is 30 Hz, *i.e.,* $\Delta t = 33.33\ ms$. The overall setup is shown in Fig. 5.

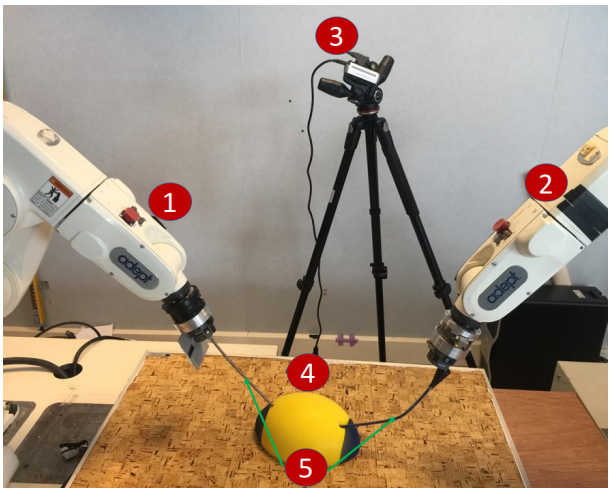


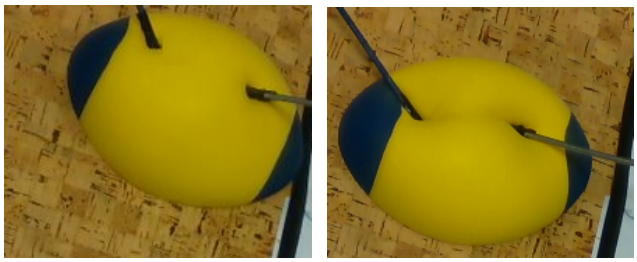
Fig. 5. 1: Viper 650. 2: Viper 850. 3: RGB-D camera. 4: Soft object. 5: Rigid sticks attached to Viper850 and Viper650.

The chosen object is an ellipsoid ball shown in Fig. 6. Its MSM parameters are approximated as presented in [1]. More precisely, we set $\boldsymbol{K}_{ij} = \text{diag}(30, 10, 30)\ N.m^{-1}, \forall i \in \mathcal{N}$, $j \in \boldsymbol{\nu}_i$ and $D_v = 0.35\ N.s.m^{-1}$. Since both the model and its parameters are coarse, a drift may appear between the real dynamical behavior of the object and its model. To get rid of this drift, the object is segmented and tracked each time a new image is acquired using the BGSLibrary [16]. The object surface points $\boldsymbol{x}_o$ consist of two components: one that is in contact with the table, and the other $\boldsymbol{x}_s \subset \boldsymbol{x}_o$ that is observable from the camera. For every surface point of the model with coordinates $\boldsymbol{x}_{s_i}$, its corresponding point $\hat{\boldsymbol{p}}_{s_i}$ on the real object surface is obtained thanks to an iterative closest point (ICP) procedure [1], [17]. A drift is detected when these two point clouds are no longer overlapping. Finally, for compensating this drift, external forces $\boldsymbol{f_c}_{s_i} = \boldsymbol{K}_{ij}(\hat{\boldsymbol{p}}^{s_i} - \boldsymbol{x}_{s_i})$ are set in (1) for all the observable surface points (see Section II-A) so that all points in the model are deformed to comply with these constraints, as proposed in [1]. As a result, this correction phase brings the model very close to the real object and we use the former for the calculation of the features $\boldsymbol{s}$ given in (5). We selected $p = 1$ and $q = 3$ harmonics in $\boldsymbol{s}$ since this choice enables to correctly represent the object without being excessively under-actuated ($k = 24$ while 6 DOF are available with $M = 2$ manipulated points).

B. Experimental results

Two experiments are described in this paper to validate our approach, while additional results are presented in the accompanying video. The convergence is considered achieved as soon as the error $\|\boldsymbol{s} - \boldsymbol{s}^*\|$ reaches 5% of its initial value.

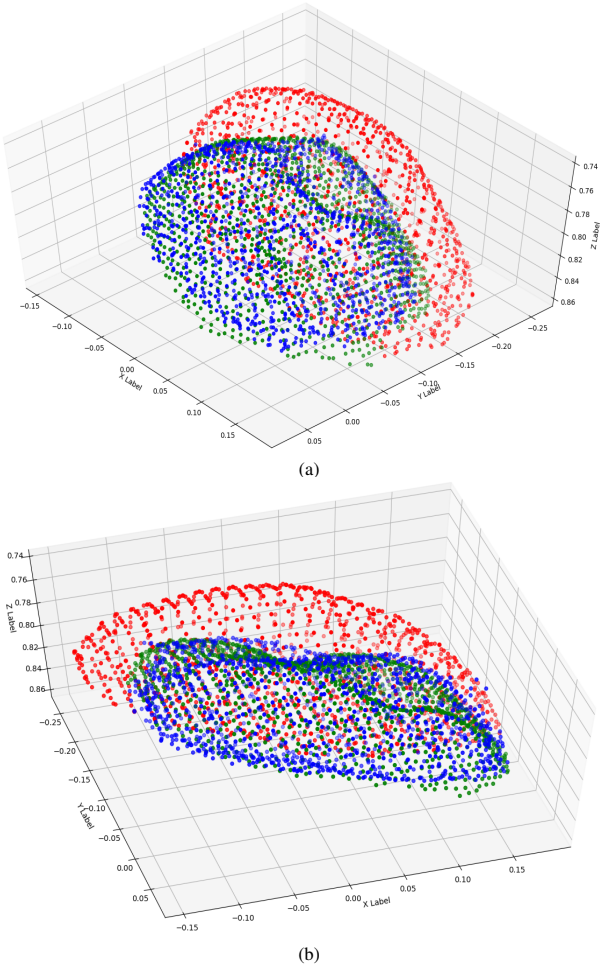
For the first experiment, Fig. 6(a) shows the object at its initial configuration while Fig. 6(b) displays the object after convergence to its desired configuration. Fig. 7 displays a



(a) (b)

Fig. 6. First experiment: initial (a) and final (b) images acquired by the RGB-D camera.

3D representation of the object surface points observed from two different viewpoints (initial configuration in red, final and desired configurations respectively in green and blue).

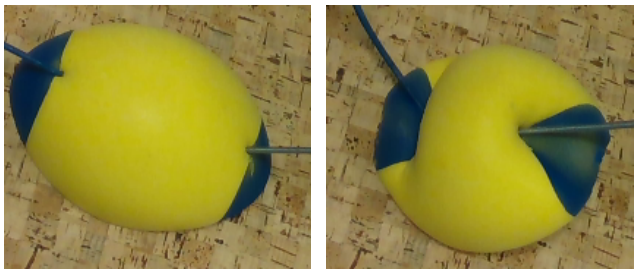


(a)

(b)

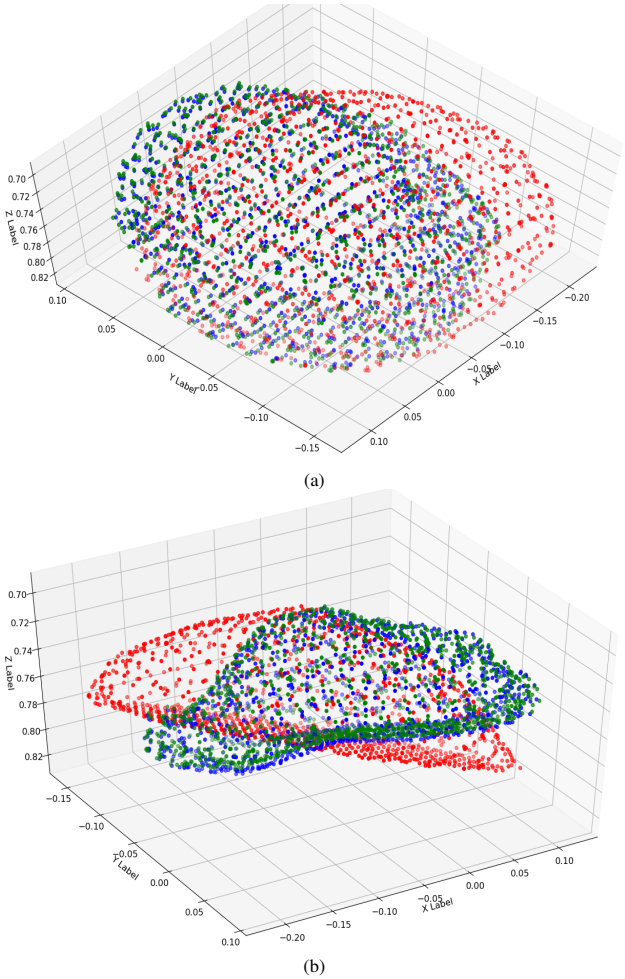
Fig. 7. First experiment: 3D visualization from 2 different viewpoints of the object surface points (initial positions in red, final positions in green, and desired positions in blue).

The second experiment involves a more complex deformation, as can be seen on Figs. 8 and 9.



(a) (b)

Fig. 8. Second experiment: initial and final images acquired by the RGB-D camera.



(a)

(b)

Fig. 9. 3D visualization from 2 different viewpoints for the second experiment.

For both experiments, the time evolution of the error norm $\|\boldsymbol{s} - \boldsymbol{s}^*\|$ is shown in Fig. 10, while Fig. 11 presents the evolution of the Hausdorff distance between the current and desired surface points.

The correct achievement of the deformation task is directly visible from Figs. 7 and 9 where we can note that the final shapes (in green) are well superposed on the desired ones (in blue) despite the displacements and deformations to be achieved. For both medium and large deformations, we can

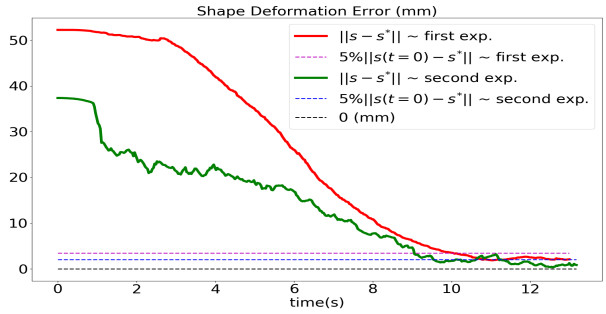


Fig. 10. Evolution of $\|\boldsymbol{s} - \boldsymbol{s}^*\|$ (first experiment in red, second one in green).

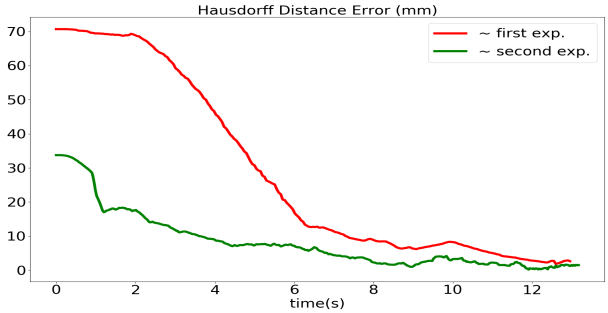


Fig. 11. Evolution of the Hausdorff distance between the current and desired object surface points (first experiment in red, second one in green).

see that the features error and Hausdorff distance converge to less than 5% of their initial value in few seconds while we recall that the system is under-actuated. The decreasing is more erratic in the case of the large deformation, which is not surprising for this difficult case. Notice that the initial errors are smaller in this case just because the rigid displacement is smaller. Finally, these results confirm that the selected harmonics were adequate for approximating the object shape and performing the required deformations.

V. Conclusion

In this paper, a new physics-based control scheme was introduced for shaping soft objects into desired configurations using a RGB-D camera and multiple manipulated points. The shape of the object, whether it is planar or volumetric, is represented by a selected number of Fourier coefficients approximating respectively its surface or contour. The relationship between the variation of these parameters and the motion of the robotic manipulators was established using a coarse mass-spring model. This relation allowed us to design the controller that contains a feedback and a feed-forward term. To get rid of the model approximation, a tracking step was performed to cancel the difference between this model and the observed shapes. The proposed approach was evaluated through simulations and real experiments, yielding promising results without the need for special markers or textures. In contrast to existing methods that were limited to servo only the 2D contour of planar or volumetric objects, the proposed approach allows deforming the complete shape of the object by considering its entire surface. As future work, we aim to explore alternative methods of parameterization to represent soft object shapes.

References

[1] F. Makiyeh, M. Marchal, F. Chaumette, and A. Krupa, "Indirect positioning of a 3d point on a soft object using rgb-d visual servoing and a mass-spring model," in *Int. Conf. on Control, Automation, Robotics and Vision (ICARCV'22)*, 2022, pp. 235–242.

[2] C. Collewet and F. Chaumette, "A contour approach for image-based control on objects with complex shape," in *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS'2000)*, vol. 1, 2000, pp. 751–756.

[3] F. Chaumette, "Image moments: a general and useful set of features for visual servoing," *IEEE Trans. on Robotics*, vol. 20, no. 4, pp. 713–723, 2004.

[4] A. Y. Yazicioglu, B. Calli, and M. Unel, "Image based visual servoing using algebraic curves applied to shape alignment," in *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS'09)*, 2009, pp. 5444–5449.

[5] D. Navarro-Alarcon and Y.-H. Liu, "Fourier-based shape servoing: A new feedback method to actively deform soft objects into desired 2-d image contours," *IEEE Trans. on Robotics*, vol. 34, no. 1, pp. 272–279, 2018.

[6] J. Qi, G. Ma, J. Zhu, P. Zhou, Y. Lyu, H. Zhang, and D. Navarro-Alarcon, "Contour moments based manipulation of composite rigid-deformable objects with finite time model estimation and shape/position control," *IEEE/ASME Trans. on Mechatronics*, vol. 27, no. 5, pp. 2985–2996, 2022.

[7] 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. on Robotics*, vol. 29, no. 6, pp. 1457–1468, 2013.

[8] D. Navarro-Alarcon, Y.-h. Liu, J. G. Romero, and P. Li, "On the visual deformation servoing of compliant objects: Uncalibrated control methods and experiments," *The Int. Journal of Robotics Research*, vol. 33, no. 11, pp. 1462–1480, 2014.

[9] J. Zhu, D. Navarro-Alarcon, R. Passama, and A. Cherubini, "Vision-based manipulation of deformable and rigid objects using subspace projections of 2d contours," *Robotics and Autonomous Systems*, vol. 142, p. 103798, 2021.

[10] M. Shetab-Bushehri, M. Aranda, Y. Mezouar, and E. Özgür, "As-rigid-as-possible shape servoing," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 3898–3905, 2022.

[11] F. Ficuciello, A. Migliozzi, E. Coevoet, A. Petit, and C. Duriez, "Fem-based deformation control for dexterous manipulation of 3d soft objects," in *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS'18)*, 2018, pp. 4007–4013.

[12] J. Das and N. Sarkar, "Autonomous shape control of a deformable object by multiple manipulators," *Journal of Intelligent & Robotic Systems*, vol. 62, no. 1, pp. 3–27, 2011.

[13] ——, "Passivity-based target manipulation inside a deformable object by a robotic system with noncollocated feedback," *Advanced Robotics*, vol. 27, no. 11, pp. 861–875, 2013.

[14] F. Chaumette and S. Hutchinson, "Visual servo control. i. basic approaches," *IEEE Robotics & Automation Magazine*, vol. 13, no. 4, pp. 82–90, 2006.

[15] A. A. Taha and A. Hanbury, "An efficient algorithm for calculating the exact hausdorff distance," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 37, no. 11, pp. 2153–2163, 2015.

[16] A. Sobral and T. Bouwmans, "Bgs library: A library framework for algorithm's evaluation in foreground/background segmentation," in *Handbook on "Background Modeling and Foreground Detection for Video Surveillance", Chapter 23*. Taylor & Francis, 2014.

[17] A. Petit, V. Lippiello, G. A. Fontanelli, and B. Siciliano, "Tracking elastic deformable objects with an rgb-d sensor for a pizza chef robot," *Robotics and Autonomous Systems*, vol. 88, pp. 187–201, 2017.