Indirect Positioning of a 3D Point on a Soft Object Using RGB-D Visual Servoing and a Mass-Spring Model

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

***Abstract*— In this paper, we present a complete pipeline for positioning a feature point of a soft object to a desired 3D position, by acting on a different manipulation point using a robotic manipulator. For that purpose, the analytic relation between the feature point displacement and the robot motion is derived using a coarse mass-spring model (MSM), while taking into consideration the propagation delay introduced by a MSM. From this modeling step, a novel closed-loop controller is designed for performing the positioning task. To get rid of the model approximations, the object is tracked in real-time using a RGB-D sensor, thus allowing to correct on-line any drift between the object and its model. Our model-based and vision-based controller is validated in real experiments for two different soft objects and the results show promising performance in terms of accuracy, efficiency and robustness.**

I. INTRODUCTION

Robotic manipulation of rigid objects has been studied and successfully implemented in several traditional application fields. However, the manipulation of deformable objects is still an open problem since the object response to an external mechanical action cannot be predicted in the same simple manner as for rigid objects. Manipulating a cable, tearing a 2D cloth, controlling the 3D shape of biological tissues are some examples of challenging soft object manipulations. The automation of these tasks combines many areas such as control, perception and modeling/learning to name a few, where several issues complicate it: i) the material properties of the object, such as its stiffness, mass and viscosity, ii) the object geometry (dimension and shape), iii) the forces to be applied (localization, direction and magnitude).

In this work, we focus on the indirect positioning of a soft object where a feature point of the latter is driven to a desired position by acting on another point of the object. Some pioneering works in the literature have already focused on this task. The reader can refer to Wada et al. who introduced the problem of indirect simultaneous positioning for soft objects [1]. They further developed a control law based on a PID controller and a MSM, which has been validated in simulation for small deformations [2]. In addition, Shibata et al. studied how to tune the parameters of this PID controller for the indirect positioning of a point in a 1D linear spring [3]. Kinio et al. used a H∞ controller and modeled the object using the Finite Element Method (FEM) to perform the positioning task [4]. Other methods that do not depend on physical models are based on the numerical estimation of a deformation Jacobian [5]–[7] that relates the feature point motion to the manipulated point motion. The main limitations of these model-free methods are their sensitivity to measurement noise and their inability to guarantee if the desired deformation can be reached in advance. In case the deformation is not feasible, the soft object can be destroyed, which is a main concern for some applications such as the robot-assisted surgery. On the contrary, a physical model of the object could be used to predict the feasibility of its deformation.

A physical model is used to mimic the actual behavior of the object under its exposure to any external force. This model does not consider only the object geometric shape that is generally described by a 3D mesh, but also its mechanical properties. In order to simulate the object dynamics, it is required to update the position of every point in the model by solving a Partial Differential Equation (PDE). Many approaches can be used, as for example the FEM, which is a continuum-based method and one of the most popular to solve PDE on irregular grid [8]. Another more simple model is the Mass-Spring Model (MSM), where the object is discretized with masses connected with springs. In this paper, MSM is chosen because it is simple to build and provides real-time capability.

A main challenge when working with physical models is to know the material properties of the object, such as stiffness for MSM. These parameters can be identified from physical interactions with the object. Since the identification of these parameters is not our main concern, we rely on the approach proposed in [9] for providing a rough approximation of the object stiffness. However, using coarse parameters prevents the MSM from imitating perfectly complex deformations and therefore leads the model to drift from the reality. Combining visual tracking with a coarse physical model allows reducing those limitations. That is why in our presented solution, the real object is tracked in each frame of a RGB-D camera using a similar method as in [10]. In case a drift occurs, an external constraint on the object surface is applied to the model for compensating it.

This paper presents a vision-based and model-based deformation control of soft objects whose main contributions are:

- A new analytical form of the relation between the motion of a feature point and the motion of a manipulated point is derived. This relation is based on a MSM taking into consideration its propagation delay. It is developed for two cases: a static manipulation and a dynamic one. For the static manipulation case, the

*This work was supported by the GentleMAN (299757) and the BIFROST (313870) projects funded by The Research Council of Norway.

[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

relation is developed for a simple 1D mesh and the model is considered in its equilibrium state before a new displacement is applied by the manipulator. This assumption is not considered in the case of dynamic manipulation, which is used in practice, and the relation is directly developed for a general 3D mesh.

- A novel control law is designed for indirectly positioning a feature point to a 3D desired position, by acting on a distant point. This control law contains a feedback term and a feedforward term whose values are computed from the modeling step, while previous approaches only considered a numerically-estimated feedback term [2], [5], [6].

An overview of our approach is presented in Fig. 1. The

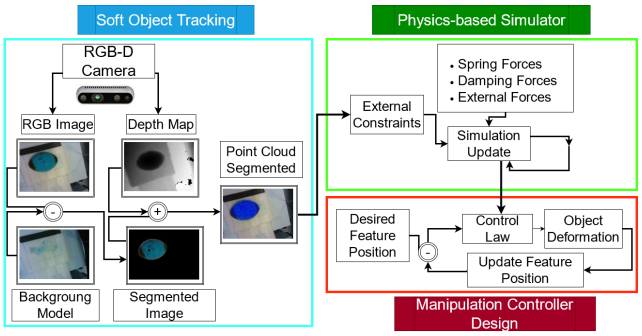


Fig. 1. Block diagram of the dynamic manipulation with visual tracking part in blue, physical simulator in green, and closed-loop control scheme in red.

main steps can be outlined as:

1) Compute the velocity to be applied to the manipulated point (see Section III).
2) Update the physical model using the semi-implicit Euler equations (see Section II-A).
3) Reduce the gap between the physical model and the real object by applying external constraints to the model surface from observations provided by the RGB-D camera (see Section IV-B).
4) Compute the error between the 3D position of the feature point and its desired position (see Section III). If this error is not low enough, go to step 1).

Note that steps 1, 3 and 4 run at the camera frame rate, i.e., 30 Hz, while step 2 runs at 600 Hz. In what follows, we use dt to denote the model update step time and Δt the step time of the camera and the control law.

The experimental validation of our method is presented in Section IV. It involves a robot manipulating two different objects with different geometries and materials.

II. Deformable Object Modeling

The proposed controller is based on a dynamic model of the object. Hence, Section II-A reviews the mass-spring model that we selected. Then, Section II-B describes how its parameters have been determined.

A. MSM dynamic behavior

A MSM approximates the behavior of a deformable object by the dynamics of a set of masses (points), linked by massless springs with dampers as connections to create a volumetric mesh. We consider a general mass-spring system, where a mass point P_i is linked to finite neighbors P_j with springs of rest length l_{ij}^0 and stiffness $\mathbf{K}_{ij} = \text{diag}(k_x, k_y, k_z)$, where (k_x, k_y, k_z) represent respectively the spring stiffness along the three axes $(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z})$. Using Newton's law, the dynamic behavior of point P_i is described as:

$$m_i \ddot{\boldsymbol{x}}_i = \boldsymbol{f_{s}}_i + \boldsymbol{f_{D}}_i + \boldsymbol{f_{G}}_i + \boldsymbol{f_{g}}_i + \boldsymbol{f_{c}}_i = \boldsymbol{f}_i. \tag{1}$$

where:

- m_i is the mass of point P_i where $i \in \mathcal{N} = [1, ..., N]$ and N is the number of points in the mesh;
- $(\boldsymbol{x}_i, \dot{\boldsymbol{x}}_i = \boldsymbol{v}_i, \ddot{\boldsymbol{x}}_i = \mathbf{a}_i)$ are respectively the 3D coordinates of P_i, its velocity and its acceleration;
- $\boldsymbol{f_{s}}_i$ is the 3D force vector acting on P_i due to springs with stiffness $\mathbf{K}_{ij}$ connecting P_i to its neighbors, P_j, $\forall j \in \boldsymbol{\nu}_i \subset \mathcal{N}$, given by:

$$\begin{aligned} \boldsymbol{f_{s}}_i &= \sum_{j \in \boldsymbol{\nu}_i} \boldsymbol{f_{s}}_{ij} = \sum_{j \in \boldsymbol{\nu}_i} \mathbf{K}_{ij} (\|\boldsymbol{x}_i - \boldsymbol{x}_j\| - l_{ij}^0) \frac{(\boldsymbol{x}_j - \boldsymbol{x}_i)}{\|\boldsymbol{x}_j - \boldsymbol{x}_i\|} \\ &= \sum_{j \in \boldsymbol{\nu}_i} \alpha_{ij} \mathbf{K}_{ij} (\boldsymbol{x}_j - \boldsymbol{x}_i) \end{aligned} \tag{2}$$

  where $\alpha_{ij} = \frac{\|\boldsymbol{x}_i - \boldsymbol{x}_j\| - l_{ij}^0}{\|\boldsymbol{x}_j - \boldsymbol{x}_i\|}$, and $\boldsymbol{\nu}_i$ denotes the indices in $\mathcal{N}$ of the points in the neighborhood of P_i.
- $\boldsymbol{f_{D}}_i = -D_v \boldsymbol{v}_i$ is the 3D force vector acting on P_i due to the damping D_v;
- $\boldsymbol{f_{c}}_i$ is the 3D force vector representing additional external constraints. This force is activated each time a new image is acquired for correcting the drift between the model and the real object. It will be discussed in more details in Section IV-B.
- $(\boldsymbol{f_{G}}_i, \boldsymbol{f_{g}}_i, \boldsymbol{f}_i)$ are respectively the 3D gravitational force, the normal force induced by the support on which the object lies, and the summation of all forces exerted on P_i.

The mechanical behavior of the object described by (1) must be solved for updating the position of the points at each time t. We use the semi-implicit Euler integration [11], also called backward integration, for its simplicity and computational efficiency [12]:

$$\dot{\boldsymbol{x}}_i^{t+dt} = \dot{\boldsymbol{x}}_i^t + \frac{\boldsymbol{f_i}^t}{m_i} dt \tag{3}$$

$$\boldsymbol{x_i}^{t+dt} = \boldsymbol{x_i}^t + \dot{\boldsymbol{x}}_i^{t+dt} dt \tag{4}$$

B. MSM parameters identification

Regarding the model parameters, some of them are fixed by the object mesh, such as point coordinates $\boldsymbol{x_i}^{t=0}$ and length of springs l_{ij}^0, $\forall i \in \mathcal{N}$, $j \in \boldsymbol{\nu}_i$ at rest state. Assuming the objects are homogeneous for dealing with the simplest model, the mass of the object is equally distributed on all points, *i.e.*, $m_i = m_1, \forall i \in \mathcal{N}$.

Concerning the spring stiffness $\mathbf{K}_{ij}$, Lloyd et al. derived an analytical expression of the stiffness of every tetrahedron constituting a mesh [9]. It is proportional to the volume of the tetrahedron and the Young's modulus. The latter is

considered constant. Furthermore, during the object deformation, the volume of the tetrahedra constituting the mesh changes and consequently the stiffnesses of the springs. Based on [13], the bigger stiffness corresponding to the biggest volume that could appear is chosen to be the same for all springs in order to prevent collisions between mesh points. The stiffness matrix $\mathbf{K}_{ij}$ therefore depends on three constant values (k_x, k_y, k_z) while k_{max} is the maximum value in $\mathbf{K}_{ij}$. An additional constraint concerning the stiffness is that $k_{max} < \frac{m_1}{\pi^2 dt^2}$ [12].

Regarding the damping value, it is calculated as presented in [12] in order to guarantee the numerical stability of the system: $2\sqrt{m_1 k_{max}} \leq D_v \leq \frac{\|\dot{\boldsymbol{x}}_i^t \frac{m_1}{dt} + \boldsymbol{f}_i{}^t\|}{\|\dot{\boldsymbol{x}}_i{}^t\|}$ for $\|\dot{\boldsymbol{x}}_i{}^t\| \neq 0$ and $D_v = 0$ when $\|\dot{\boldsymbol{x}}_i{}^t\| = 0$ since the damping has no effect on a static point. In this work, we choose $D_v = 2\sqrt{m_1 k_{max}}$.

Despite the simplicity of the selected model parameters, which allows high-speed real-time capacities, we will see that it is possible to consider large deformations on real objects thanks to the proposed closed-loop strategy, both in terms of control and model/object registration.

III. Manipulation controller design

In this section we present the main contribution of this paper that concerns the generation of velocities to be applied to a manipulated point $\boldsymbol{x}_m$ to drive automatically another point $\boldsymbol{x}_f$ to a desired position $\boldsymbol{x}_f^{\text{des}}$. Note that our velocity-based controller has 3 Degrees of freedom (Dof) since it acts on the position of point $\boldsymbol{x}_m$. We first formulate the displacements of the mesh points when an external displacement Δ_m is imposed on $\boldsymbol{x}_m$, using an example of four particles along a single axis (for explanation purpose) as represented in Fig. 2. Then we derive in Section III-B this relation for a general 3D mesh model using a series of displacement instead of just one.

We recall that the dynamics of the mesh points follows (1) and their positions are updated using (3) and (4). At time $t+dt$, we then obtain for any point P_i, $i \in \mathcal{N}-\{m\}$:

$$\boldsymbol{x}_i{}^{t+dt} = \boldsymbol{x}_i{}^t + \frac{dt^2}{m_i}\boldsymbol{f_s}_i^t + (dt - \frac{dt^2}{m_i}D_v)\dot{\boldsymbol{x}}_i^t + \frac{dt^2}{m_i}\boldsymbol{f_e}_i \quad (5)$$

with $\boldsymbol{f_e}_i = \boldsymbol{f_G}_i + \boldsymbol{f_g}_i + \boldsymbol{f_c}_i$. For the manipulated point P_m, its position $\boldsymbol{x}_m$ is imposed by the robotic manipulator since it is attached to it. Moreover, we consider that the forces $\boldsymbol{f_s}$ and $\boldsymbol{f_D}$ are the dominant effects for the deformation, because they depend on the model parameters (stiffness and damping). Hence, we neglect the effects of the external forces $\boldsymbol{f_e}$ from the control modeling by considering their effects as small.

Point P_1 in Fig. 2 is considered as the manipulated point. It is subject to a displacement $\Delta_1 < l_{12}^0$. P_2 and P_3 are free to move while P_4 is a fixed point. At time t_0, these particles are considered at rest state. The displacement Δ_1 of P_1 propagates first to its neighbor and then to the neighbor of its neighbor, until finally reaching the farthest point. Let γ_i^t be the propagation coefficient variable that denotes the displacement ratio of P_i with respect to Δ_1 at every discrete time t. At $t_1 = t_0+dt$, only P_2 moves and $x_2^{t_1} = x_2^{t_0} + \gamma_2^{t_1}\Delta_1$, due to the propagation delay introduced by the MSM. At $t \geq t_2 = t_0 + 2dt$, we have $x_2^t = x_2^{t_0} + \gamma_2^t\Delta_1$, $x_3^t = x_3^{t_0} + \gamma_3^t\Delta_1$ and $x_4^t = x_4^{t_0}$. Using (5), we can find γ_i^t at each time t:

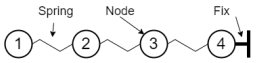


Fig. 2. An example of 1D MSM with 4 nodes.

1) $\gamma_2^t = \gamma_2^{t-dt} - K_x\frac{dt^2}{m_1}(\gamma_2^{t-dt} - \gamma_3^{t-dt}) + K_x\frac{dt^2}{m_1}(1 - \gamma_2^{t-dt}) + (1 - \frac{dt}{m_1}D_v)(\gamma_2^{t-dt} - \gamma_2^{t-2dt})$ for $t \geq t_0 + 2dt$, with $\gamma_2^{t_0+dt} = K_x\frac{dt^2}{m_1}$.
2) $\gamma_3^t = \gamma_3^{t-dt} - K_x\frac{dt^2}{m_1}(\gamma_3^{t-dt}) + K_x\frac{dt^2}{m_1}(\gamma_2^{t-dt} - \gamma_3^{t-dt}) + (1 - \frac{dt}{m_1}D_v)(\gamma_3^{t-dt} - \gamma_3^{t-2dt})$ for $t \geq t_0 + 3dt$, with $\gamma_3^{t_0+2dt} = K_x\frac{dt^2}{m_1}\gamma_2^{t_0+dt}$ and $\gamma_3^{t_0+dt} = 0$.

Using the same methodology for a complete 3D mesh, it is possible to determine $\boldsymbol{\gamma}_i$, a 3D propagation matrix, for any point of the mesh (for the fixed point, we have of course $\boldsymbol{\gamma}_i = 0$).

We now propose to derive the analytical form of the relation that gives the displacement of a feature point of a soft object in function of the motion of a manipulated point. This relation is derived for two cases. Static manipulation refers to the case where a new control displacement is applied on the manipulated point only once the MSM reached its equilibrium state from the previous control displacement. This means that a large number of iterations of the MSM model are required to obtain its equilibrium state before applying the next control motion. In contrast, the dynamic manipulation case does not require the MSM to reach its equilibrium before applying the next control motion and has therefore the advantage to take into account the transient dynamics of the model.

A. Static manipulation

For the static manipulation case, we again take the simple model in Fig. 2 and consider that a displacement is imposed on P_1 at t_0, but we wait for the model to reach a new equilibrium state at t_∞. After multiple iterations, the propagation variable for each free point γ_i^t converges to a nonzero stable value $\gamma_i^{t_\infty}$. This convergence is guaranteed by the choice of the model parameters. As discussed in Section II-B, these parameters were chosen for ensuring the numerical stability of the system subject to internal and external forces, thus avoiding oscillations and divergence. As a result, the convergence of γ_i is asserted. Hence, for the simple mesh presented in Fig. 2, once a displacement Δ_1 is applied on P_1, P_2 and P_3 move respectively by $x_2^{t_\infty} - x_2^{t_0} = \gamma_2^{t_\infty}\Delta_1$ and $x_3^{t_\infty} - x_3^{t_0} = \gamma_3^{t_\infty}\Delta_1$.

Finally, in order to perform an indirect positioning of P_3 to a desired position x_3^{des}, a simple strategy is to apply the displacement Δ_1 on P_1 given by:

$$\Delta_1 = -(\gamma_3^{t_\infty})^{-1}(x_3^{t_0} - x_3^{des}) \quad (6)$$

$\gamma_3^{t_\infty}$ in (6) is a nonzero scalar whose calculation was derived above. In this case, the displacement Δ_1 is just given by a proportional gain, $(\gamma_3^{t_\infty})^{-1}$, multiplied by the initial error $(x_3^{des} - x_3^{t_0})$ of the positioning task. This result is similar to

the use of a simple PID controller as proposed in [2], [3]. We will further discuss this approach in Section IV-C.

We can already notice that if the desired position is close to P_3, P_3 will reach this desired position. However, if the desired position is far from P_3, then a large Δ_1 may be required. A large external displacement can induce the divergence of the model or the destruction of the object, so a succession of smaller motions have to be applied on P_1 instead of a large one. We recall that, from the previous developments, it is necessary to wait until the model reaches its stationary state before applying a new displacement. That is why, in the following section, we are interested in the application of successive movements without waiting for the system to reach its new equilibrium state between each of them.

B. Dynamic manipulation

For the dynamic manipulation case, a general 3D mesh is considered and the displacement on P_m is expressed as a series of displacements with a frequency of 30 Hz. Let us denote b as a positive integer number. The displacement applied on P_m at time $t = b\Delta t$ is given by $\boldsymbol{\Delta}_m^{(b+1)\Delta t} = \Delta t\, \dot{\boldsymbol{x}}_m^{(b+1)\Delta t}$. It is considered constant for all time t such that $b\Delta t \leq t < (b+1)\Delta t$. Moreover, we use the left superscript (ind) to denote that the corresponding vector is independent of $\boldsymbol{\Delta}_m^{(b+1)\Delta t}$ between $b\Delta t \leq t < (b+1)\Delta t$.

By integrating (5) between $b\Delta t$ and $(b+1)\Delta t$, it results that the displacement of any point P_f due to P_m motion is given by:

$$\boldsymbol{x}_f^{(b+1)\Delta t} = \boldsymbol{x}_f^{b\Delta t} + \boldsymbol{\gamma}_f^{(b+1)\Delta t} \Delta t\, \dot{\boldsymbol{x}}_m^{(b+1)\Delta t} + {}^{\text{ind}}\boldsymbol{r}_f^{(b+1)\Delta t} \quad (7)$$

The intermediate equations used to obtain (7) are developed in the Appendix.

It has to be noted that $\boldsymbol{\gamma}_f^{(b+1)\Delta t}$ has to be calculated for each new displacement on P_m. Its value is given by (21) and denotes the propagation ratio of the motion on P_f imposed by P_m between $b\Delta t$ and $(b+1)\Delta t$. The term ${}^{\text{ind}}\boldsymbol{r}_f{}^{(b+1)\Delta t}$ is calculated using (22). It combines the effects of the spring and damping forces exerted on P_f between $b\Delta t$ and $(b+1)\Delta t$, regardless the new motion exerted on P_m at $b\Delta t$. Recall that $\boldsymbol{\gamma}_f^{(b+1)\Delta t} \neq \mathbf{0}$ in (7). This means that the displacement on P_m is propagated to all mesh points, and specifically to P_f, even if it is far away from P_m. This assumption is valid because the model is updated using (5) at 600 Hz, providing a sufficient number of iterations for which the displacement of P_m propagates to all model points before a new one takes place.

Finally, by using $\dot{\boldsymbol{x}}_f^{(b+1)\Delta t} = \frac{\boldsymbol{x}_f^{(b+1)\Delta t} - \boldsymbol{x}_f^{b\Delta t}}{\Delta t}$ in (7), we obtain:

$$\dot{\boldsymbol{x}}_f^{(b+1)\Delta t} = \boldsymbol{\gamma}_f^{(b+1)\Delta t}\ \dot{\boldsymbol{x}}_m^{(b+1)\Delta t} + \frac{{}^{\text{ind}}\boldsymbol{r}_f^{(b+1)\Delta t}}{\Delta t} \quad (8)$$

The term $\boldsymbol{\gamma}_f^{(b+1)\Delta t}$ is equivalent to the numerically-estimated deformation Jacobian used in the model-free approaches [5], [6], while in this work we analytically derive its value. Note that these model-free approaches do not consider the second term ${}^{\text{ind}}\boldsymbol{r}_f^{(b+1)\Delta t}/\Delta t$, which represents the transient dynamics of the system. This leads to a proportional-like controller, while the one we propose contains also a feedforward term. Indeed, in order to drive the feature point P_f to its desired position $\boldsymbol{x}_f^{\text{des}}$ with an exponential decoupled decrease, a closed-loop can be performed by imposing $\dot{\boldsymbol{x}}_f{}^{(b+1)\Delta t} = -G_p \boldsymbol{e}^{b\Delta t}$, with $\boldsymbol{e}^{b\Delta t} = \boldsymbol{x}_f{}^{b\Delta t} - \boldsymbol{x}_f^{\text{des}}$ the positioning error and G_p a proportional positive gain. By identification with (8), the closed-loop control law to be applied by the robot on P_m is given by:

$$\dot{\boldsymbol{x}}_m^{(b+1)\Delta t} = -(\boldsymbol{\gamma}_f{}^{(b+1)\Delta t})^{\dagger} (G_p \boldsymbol{e}^{b\Delta t} + \frac{{}^{\text{ind}}\boldsymbol{r}_f^{(b+1)\Delta t}}{\Delta t}) \quad (9)$$

with $(\boldsymbol{\gamma}_f{}^{(b+1)\Delta t})^{\dagger}$ the pseudo-inverse of $\boldsymbol{\gamma}_f^{(b+1)\Delta t}$ in case its rank is less than 3. Otherwise, $(\boldsymbol{\gamma}_f{}^{(b+1)\Delta t})^{\dagger} = (\boldsymbol{\gamma}_f^{(b+1)\Delta t})^{-1}$.

Eq. (9) presents an exclusive relation between the error of the positioning task and the manipulated point velocity at time $(b+1)\Delta t$ according to the model parameters. All terms at the right hand side of (9) can be calculated by updating the model between $b\Delta t$ and $(b+1)\Delta t$ regardless $\boldsymbol{\Delta}_m^{(b+1)\Delta t}$ (see more details in the Appendix).

The validation and robustness of this novel control law is analyzed in the next section through experimental results. Concerning the complexity, since we rely on a MSM, only a low computational cost is required, which allows the control law to run efficiently in real-time.

IV. Experimental results

A. Setup & Implementation

Our experimental setup is composed of a Viper 850, 6-Dof anthropomorphic robot arm from ADEPT, equipped with an ATI's Gamma IP65 force/torque sensor and a rigid stick used as an end-effector distal tool.

To control the robot, we use the ViSP library [14]. The visual feedback is acquired with a RGB-D Intel Realsense D435 static camera. The RGB-D camera simultaneously acquires color and depth images, with a (640x480) resolution, at 30 frames per second (fps).

A physical simulator has been developed in C++ to represent the model of the deformable object and update its point positions. The control velocity (9) is generated in the camera frame, which is different from robot base frame. That is why an eye-to-hand calibration is performed once at the start of the experiments, since our camera and the robot base frames are stationary. The overall setup is shown in Fig. 3.

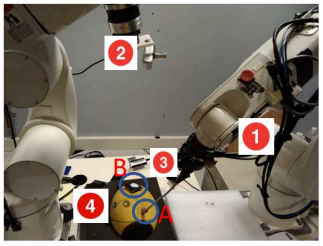


Fig. 3. 1: Viper 850. 2: RGB-D camera. 3: Rigid stick. 4: Soft object. A: Manipulation point. B: Marker which is considered as feature point.

B. Model tracking and correction using point clouds

Concerning the model representing the object, a surface mesh is first obtained from the RGB-D data using the PCL library, and then a volumetric mesh is generated. Since the object is positioned on a flat table, the part of the object lying on it is not visible, introducing a lack of measurements for this part. The PCL library provides a simple way to determine the plane on which the object lies [15]. Then, the surface points are generated by selecting the points lying inside the plane hull. A surface mesh is therefore generated by connecting these points with each other, ending up with a continuous polygon mesh constituted by triangle faces. Finally, a volumetric mesh is generated by filling the surface mesh by tetrahedra using Gmsh tool [16].

The model is then updated using (5) by taking into account the internal forces that depend on the model parameters, the known external forces that depend on the object configuration, and the manipulator point displacement. Since the MSM is an approximation of the real behavior of the object, the error between the model and the reality will be aggregated during the manipulation and the gap between them could lead to unrealistic object representation. In order to adjust the model and preventing it from drifting from reality, the model is adjusted each time the camera acquires a new image. For that, the object is first segmented using the BGSLibrary [17], an OpenCV C++ background subtraction.

Then, let us denote the model surface by $\boldsymbol{x}^s = [{\boldsymbol{x}_{s_1}}^T, ..., {\boldsymbol{x}_{s_w}}^T]^T,\ \forall\ 1 \leq i \leq w,\ s_i \in \mathcal{N}$ where w is the number of the mesh surface points and s_i is the indices of their corresponding points in $\mathcal{N}$. The segmented surface object using the RGB-D camera is denoted by $\boldsymbol{p}^s$. Hence, by tracking $\boldsymbol{x}^s$ and $\boldsymbol{p}^s$, the drift can be detected. It appears when these two point clouds are no more superimposed. To compensate the drift, the model is adjusted by rigidly aligning $\boldsymbol{x}^s$ on $\boldsymbol{p}^s$, using an iterative closest point (ICP) procedure. As a result, $\boldsymbol{p}^s$ corresponds to some points of $\boldsymbol{x}^s$ and they serve as the input displacements to $\boldsymbol{x}^s$. In [10], Petit et al. considered these displacements as external forces exerted by $\boldsymbol{p}^s$ on $\boldsymbol{x}^s$ and then integrated them to the mechanical model using co-rotational FEM. In a similar way, we use the same procedure but using the MSM. We consider the external forces as external constraints $\boldsymbol{f_c}$. For every point $\boldsymbol{x}_{s_i} \in \boldsymbol{x}^s, \forall\ 1 \leq i \leq w,\ s_i \in \mathcal{N}$, we denote by $\hat{\boldsymbol{p}}^{s_i} \in \boldsymbol{p}^s$ its correspondent point on the point cloud of the real object. In order to get this correspondence, we apply the same matching technique as in [10]. Two sets of nearest neighbor correspondences from $\boldsymbol{x}^s$ to $\boldsymbol{p}^s$ and from $\boldsymbol{p}^s$ to $\boldsymbol{x}^s$ are determined. Finally the correspondence $\hat{\boldsymbol{p}}^{s_i}$ is calculated as a trade-off between these last two correspondences, therefore $\mathbf{f_c}_{s_i} = \mathbf{K}_{ij}(\hat{\boldsymbol{p}}^{s_i} - \boldsymbol{x}_{s_i})$. For the remaining mesh points not belonging to the surface, no external constraint is applied to them, *i.e.,* $\boldsymbol{f_c} = 0$. However, the effect of the force applied on the outer points of the model is propagated to the inner points with (5). As a result, the correction phase leads the volumetric model to approach the real object deformation behavior.

Since correction is performed in sequences with a frequency of 30 Hz (camera frame rate) while the model is updated with a frequency of 600 Hz, the external constraints are considered constant until a new image is acquired, *i.e.,* for $b\Delta t \leq t < (b+1)\Delta t$.

C. Results

We assume in all experiments that the rigid stick of the robot is in contact with the deformable object at $t = 0$. Its position is used as the manipulated point of the object.

The deformation control scheme has been tested for two different objects made of different soft materials, as presented in Fig.4. To validate our approach, the 3D positional error between the desired position of the feature point and its position are measured at each time step, where our objective is to drive this error to zero. In this paper, we only present the results for the half ball due to paper length limitation. Additional results obtained with the second planar object are presented in the accompanying video[1].

We first determined the parameters of the object model as detailed previously in Section II-B. Then a coarse estimation of $\mathbf{K}_{ij} = \text{diag}(30, 10, 30)\ N.m^{-1}, \forall i \in \mathcal{N},\ j \in \boldsymbol{\nu}_i$ and $D_v = 0.35\ N.s.m^{-1}$ are used.

Many deformations were first performed in simulations for determining the possible gains to be applied to the proposed controller (9). Two main criterion were studied: system stability and time-to-convergence. The proportional gain finally used is $G_p = 0.7$. Then the controller has been validated using the real robot for different manipulated point positions and feature point positions.

One example of deformation task is presented in Fig. 5. The distance between the manipulated point and the feature point is chosen arbitrary and is about 75 mm. The object is restricted on one of its side to prevent its free 3D translational movement. The error of the positioning task is directly measured by the RGB-D camera. The measured depth contains oscillations in a range of 2 mm that is due to the depth measurement noise specific to this type of camera. Therefore, the convergence is considered when the error norm is around 1 mm. The green point in Fig. 5 presents the position of the feature point in the image. A visual marker is rigidly attached to the object at that point so that it is easily tracked using the ViSP library [14]. The white point indicates the desired position of the feature point. The yellow rectangle presents the restricted side of the object. Fig. 5.(a) and Fig. 5.(b) depict respectively the initial and final state for a positioning task example using our proposed approach (PA). The red, green, blue and black solid lines in Fig. 5.(c) represent respectively the evolution of the positioning error along $(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z})$ axes and the error norm when using the PA. The proposed controller drives the error norm within 1 mm, between the dashed magenta lines, with an initial error around 55 mm. We can notice the nice exponential decrease of this error, which validates the modeling proposed in the previous section. We can also notice the fast convergence since the desired position is reached in about 12 seconds.

We also performed different tests where the positions of the manipulated point and the feature one are close (N) and far apart (F). This distance can be represented as the

[1]https://youtu.be/QeyK73ZYFVA

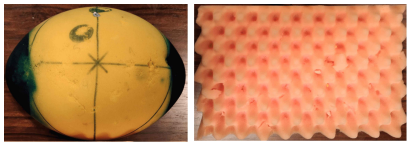


Fig. 4. Two different soft objects used to validate our approach. Left: Half of an ellipsoid ball. Right: 2D object.

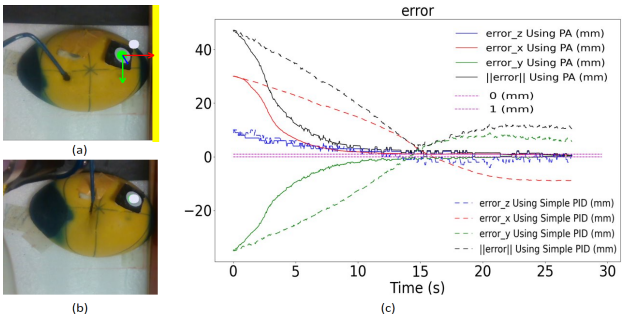


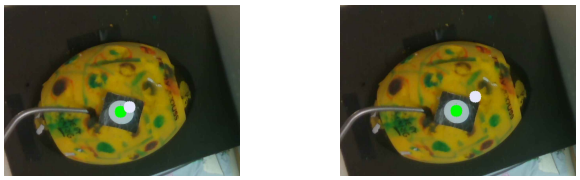
Fig. 5. (a) Soft object at its rest state. The red, green and blue oriented axes correspond to a local coordinate system with origin the initial feature point 3D position. (b) Soft object after the deformation process. (c) Error measured at each time step for the positioning task applied to the half of the ball when using our proposed control and the simple PID used in [2], [3]. The convergence of the error is assumed when its norm lies between the dashed magenta lines, 0 (mm) and 1 (mm).

distance between the center of the two blue circles A and B in Fig. 3. In addition, the controller has been tested for two types of deformations. We call small deformation (S) when the desired position of the feature point is far from its initial position by a distance less than 10% of the largest dimension of the object, *i.e.,* height, width and thickness, and large deformation (L) when this distance is between 12% to 30% of the latter. In what follows, we use some notations, such as S-N to represent small deformations and the case where the manipulated point is near to the feature point. Small deformations allow us to compare our results with the approach introduced in [2], [3]. Since their approach is independent of the mesh, their results are obtained using a simple PID controller by stating: $\dot{\boldsymbol{x}}_m = -(G_p \boldsymbol{e} + G_d \frac{\boldsymbol{\Delta e}}{\Delta t} + G_i \int \boldsymbol{e})$ with G_p, G_d and G_i being proportional, derivative and integral gains respectively.

The main problem when using this simple PID controller alone is the fine-tuning of the gains of the controller for each task to be accomplished. For a small proportional gain and depending on the soft object, the error either converged towards zero, or diverged, or converged towards a minimum other than zero. For a large proportional gain, the error either diverged, or oscillated around a local minimum, and sometimes destroyed the object. Therefore, we tried to adjust the parameters for the simple PID method by a trial and error method. Unlike this controller, our PA does not need any parameter setting except the approximate physical model parameters and the control proportional gain, which has been discussed at the beginning of this section. Additionally, since the robot used is equipped with a force sensor, the applied forces are measured during the deformation. Due to space limitations, the comparison results are summarized in Table I.

The comparison also highlights the effect of the ${}^{\text{ind}}\boldsymbol{r}_f$ and $\boldsymbol{\gamma}_f^{\dagger}$ terms in (9), which represent an important difference with respect to all previous approaches.

The table shows that when the manipulated point is near the feature point, the proposed method and the PID converge similarly without any problem. For the other cases, our method ensures the convergence of the error norm while it is not the case for the simple PID controller. The table summarizes the statistics of the different tests carried out since many experiments have been done for each case. For example, for S-N case, many manipulated positions are chosen around the feature point within the same distance. An example of the initial state for the near case is presented in Fig. 6. In addition, the measured forces clearly show that the robot applies less force to the object using the PA, which could be sometimes half and sometimes quarter of the applied force using the simple PID controller [2], [3]. The importance of this characteristic is that when working with some elastic objects, a high force can cause the object to loose its elasticity. Moreover, for the L-F case represented in Fig. 5, the error initially decreased and then diverged using the simple PID controller (see dashed lines in Fig. 5(c)).



(a): S-N, error (8,15) pixels (b): L-N, error (24,32) pixels

Fig. 6. (a), (b) represent respectively the initial position of the feature point and its desired position for the S-N and L-N cases. The figures also present the initial absolute error in terms of pixels.

Additional experiments are illustrated in Fig.7 where the variation of the error norm is presented for a L-F case, using five different configurations of the controller. The red graph presents the error norm evolution when the simple PID controller is used. We can see that the error oscillates around a local-minimum then a divergence occurs, as expected from Table. I. On the other hand, the error converges with a nice exponential decrease when our PA and the identified model parameters are used (see blue graph).

In the third configuration (see green graph), we experimentally tested the robustness of the PA against the model parameters change. They were changed arbitrary by doubling the stiffness and halving the damping. We can observe on Fig.7 that the error norm converged to zero, but with some perturbations in the exponential decrease. The black graph presents the result after multiplying by 6 the real stiffness. In that case, the system failed in a local minimum. As a conclusion, our controller is robust to a large extent to coarse model parameters. Finally, the cyan graph presents the error norm when our PA is used with a reduced mesh resolution for the object model. The object mesh used in this experiment contains 2059 nodes instead of a high mesh resolution of 7103 nodes used in the previous experiments. We can notice that such reduction induces perturbations (but no divergence)

to the system behavior, while the error still converges.

TABLE I

PA VERSUS A SIMPLE PID FOR DIFFERENT EXPERIMENTS WITH DIFFERENT INITIAL ERRORS, IN TERMS OF FINAL AVERAGE ERROR $\|\bar{\boldsymbol{e}}\|$ AND THE MAXIMAL NORM OF THE MEASURED FORCE MAX $\|\boldsymbol{F}\|$ (N).

Initial error (mm)	Proposed		Simple PID	
S-N (15,10,-5)	$\|\bar{\boldsymbol{e}}\|$ 0.7	Max $\|\boldsymbol{F}\|_2$ 2	$\|\bar{\boldsymbol{e}}\|$ 0.7	Max $\|\boldsymbol{F}\|_2$ 4
S-F (10,10,0)	$\|\bar{\boldsymbol{e}}\|$ 0.9	Max $\|\boldsymbol{F}\|_2$ 5	$\|\bar{\boldsymbol{e}}\|$ Diverged	Max $\|\boldsymbol{F}\|_2$ > 33
L-N (35,-30,10)	$\|\bar{\boldsymbol{e}}\|$ 0.9	Max $\|\boldsymbol{F}\|_2$ 10	$\|\bar{\boldsymbol{e}}\|$ 0.9	Max $\|\boldsymbol{F}\|_2$ 12.25
L-F (35,-30,10)	$\|\bar{\boldsymbol{e}}\|$ 1.1	Max $\|\boldsymbol{F}\|_2$ 13	$\|\bar{\boldsymbol{e}}\|$ Diverged	Max $\|\boldsymbol{F}\|_2$ > 30

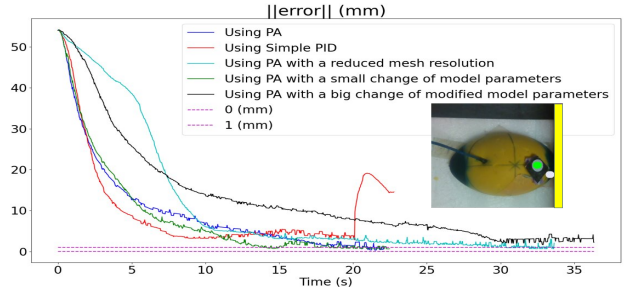


Fig. 7. Evolution of the positioning task error norm for a L-F case (illustrated on the right of the figure) and for five different controller configurations.

V. CONCLUSIONS

In this paper, we presented a physically-based method to control the 3 Dof of an industrial robot using a RGB-D camera to perform the positioning of a feature point belonging to a soft object by manipulating a distant point. We also carried out a tracking step for minimizing the gap between the coarse MSM model used and the observed deformations. The proposed approach was carefully evaluated in real experiments and appealing results were presented for small and large deformations. Moreover, the robustness of the proposed controller was experimentally tested and results showed that the approach can work with coarse model parameters and low mesh resolution. Unlike existing methods, we proposed an original approach that uses the analytical structure of the underlying system of equations. In addition, we avoided nonlinear optimization procedures, which could be computationally expensive. Finally, our approach works in real-time thanks to its low computational cost.

As future work, we will not only focus on the positioning task of a single 3D feature point but on a set of 3D feature points while manipulating multiple points. Moreover, the proposed control law (9) is general in the sense that it can consider a model in which the springs could have different stiffnesses and the points could have different masses. That is why we also would like to test our method on non-homogeneous objects, which would require to estimate more accurately its internal parameters.

APPENDIX

We demonstrate in this appendix how Eq. (7) given in Section III-B is obtained. Recall that the displacement applied on P_m between $b\Delta t$ and $(b+1)\Delta t$ is given by $\boldsymbol{\Delta}_m^{(b+1)\Delta t}$ and it is considered constant. Moreover, given the positions of all points at time $b\Delta t$, the objective is to find their positions at time $(b+1)\Delta t$. In this case, the position of any point P_i at $t = b\Delta t$ is known. For example, when $b = 0$, the position of the points corresponds to their initial position. We use the same methodology presented at the beginning of Section III. We first develop the required equation for the neighbors of the manipulated point P_m, then generate it for all points.

For any point P_M in the neighborhood of P_m, we obtain its position at $b\Delta t + dt$ using (5). We start by exploring ${\boldsymbol{f_s}}_M^t$ using (2) with $t = b\Delta t$, and to simplify the equations, we denote $\boldsymbol{d}_{ij}^t = (\boldsymbol{x}_j^t - \boldsymbol{x}_i^t)$:

$$
\begin{aligned}
{\boldsymbol{f_s}}_M^t &= \sum_{j \in \boldsymbol{\nu}_M} {\boldsymbol{f_s}}_{Mj}^t = \sum_{j \in \boldsymbol{\nu}_M} \mathbf{K}_{Mj}(\|\boldsymbol{d}_{ij}^t\| - l_{ij}^0)\frac{\boldsymbol{d}_{ij}^t}{\|\boldsymbol{d}_{ij}^t\|} \\
&= \sum_{j \in \boldsymbol{\nu}_M - \{m\}} {\boldsymbol{f_s}}_{Mj}^t + {\boldsymbol{f_s}}_{Mm}^t
\end{aligned} \tag{10}
$$

In (10), we know that $\sum_{j \in \boldsymbol{\nu}_M - \{m\}} {\boldsymbol{f_s}}_{Mj}^t$ is independent of $\boldsymbol{\Delta}_m^{(b+1)\Delta t}$ because its effect starts at $b\Delta t + dt$. Regarding ${\boldsymbol{f_s}}_{Mm}^t$, it is a function of $(\boldsymbol{x}_M^t, \boldsymbol{x}_m^t)$, *i.e.*, ${\boldsymbol{f_s}}_{Mm}^t = {\boldsymbol{f_s}}_{Mm}^t(\boldsymbol{x}_M^t, \boldsymbol{x}_m^t) = {\boldsymbol{f_s}}_{Mm}^t(\boldsymbol{x}_M^t, \boldsymbol{x}_m^{b\Delta t} + \boldsymbol{\Delta}_m^{(b+1)\Delta t})$. Using a first order approximation of ${\boldsymbol{f_s}}_{Mm}^t$, we get:

$$
{\boldsymbol{f_s}}_{Mm}^t = {\boldsymbol{f_s}}_{Mm}^t(\boldsymbol{x}_M^t, \boldsymbol{x}_m^{b\Delta t}) + \frac{\partial \boldsymbol{f_s}_{Mm}}{\partial \boldsymbol{x}_m^t}\boldsymbol{\Delta}_m^{(b+1)\Delta t} \tag{11}
$$

Moreover, we have:

$$
\begin{aligned}
\frac{\partial \boldsymbol{f_s}_{Mm}}{\partial \boldsymbol{x}_m} &= \frac{\partial \boldsymbol{f_s}_{Mm}}{\partial \boldsymbol{d}_{Mm}}\frac{\partial \boldsymbol{d}_{Mm}}{\partial \boldsymbol{x}_m} = \mathbf{K}_{Mm}\mathbf{I}_{3\text{x}3} \\
&\quad - \mathbf{K}_{Mm} l_{Mm}^0 \frac{\|\boldsymbol{d}_{Mm}\|^2 \mathbf{I}_{3\text{x}3} - \boldsymbol{d}_{Mm}\boldsymbol{d}_{Mm}^T}{\|\boldsymbol{d}_{Mm}\|^3}
\end{aligned} \tag{12}
$$

Therefore, by using (12) in (11), then in (9), we obtain:

$$
\begin{aligned}
{\boldsymbol{x}_M}^{t+dt} &= {\boldsymbol{x}_M}^t + \frac{dt^2}{m_M}\sum_{j \in \boldsymbol{\nu}_M - \{m\}} {\boldsymbol{f_s}}_{Mj}^t + (dt - \frac{dt^2}{m_M}D_v)\dot{\boldsymbol{x}}_M^t \\
&+ \frac{dt^2}{m_M}{\boldsymbol{f_s}}_{Mm}^t(\boldsymbol{x}_M^t, \boldsymbol{x}_m^{b\Delta t}) + \frac{dt^2}{m_M}\frac{\partial \boldsymbol{f_s}_{Mm}}{\partial \boldsymbol{x}_m}\boldsymbol{\Delta}_m^{(b+1)\Delta t}
\end{aligned} \tag{13}
$$

Eq. (13) shows that the last term in ${\boldsymbol{x}_M}^{t+dt}$ is the only term that depends on $\boldsymbol{\Delta}_m^{(b+1)\Delta t}$. Therefore, we propose to decouple the model point position of any point P_i into two parts:

$$
{\boldsymbol{x}_i}^t = {}^{\text{ind}}{\boldsymbol{x}_i}^t + \boldsymbol{\gamma}_i^t \boldsymbol{\Delta}_m^{(b+1)\Delta t}, \tag{14}
$$

The first part ${}^{\text{ind}}{\boldsymbol{x}_i}^t$ is independent of the motion $\boldsymbol{\Delta}_m^{(b+1)\Delta t}$ imposed on P_m between $b\Delta t < t \leq (b+1)\Delta t$. The second part $\boldsymbol{\gamma}_i^t$ represents the effect of the latter. In what follows, the left superscript (ind) is used to denote that the corresponding vector is independent of $\boldsymbol{\Delta}_m^{(b+1)\Delta t}$ between $b\Delta t \leq t < (b+1)\Delta t$.

It has to be noted that the effect of $\mathbf{\Delta}_m^{(b+1)\Delta t}$ on P_i will not take place instantaneously at $t = b\Delta t$, but at t'_i with $b\Delta t \leq t'_i < (b+1)\Delta t$. It depends on the position of P_i with respect to P_m. Then, for all $t < t'_i$, we have $\boldsymbol{\gamma}_i^t = \mathbf{0}$. Finding t'_i is an easy problem once we know P_m. The mesh can be considered as a graph, with P_m as a root and its neighbors as the leaves. Then, t'_i depends on the number of layers between the root and P_i. If this number is $r-1$, then $t'_i = b\Delta t + rdt$. For the example presented in Fig.2, we have $b = 0$ and P_2 is directly attached to P_m, then $r = 1$, $t'_2 = dt$. For P_3, there is one intermediate layer between P_1 and P_3 then $r = 2$, *i.e.*, $t'_3 = 2dt$. Returning to t, for the points P_i that they are not in the neighborhood of P_m, $\boldsymbol{\gamma}_i^t = \mathbf{0}$ in (14) at $t = b\Delta t + dt$. We suppose that the proposed decoupling for any P_i concerned in (14) is true at time t, $b\Delta t \leq t < (b+1)\Delta t$. Then, we will prove it by mathematical induction, by showing its validity at $t + dt$. Using (14), we obtain:

$$\dot{\boldsymbol{x}}_i^t = \frac{{\boldsymbol{x}_i}^t - {\boldsymbol{x}_i}^{t-dt}}{dt} = \frac{({}^{\text{ind}}\boldsymbol{x}_i^t - {}^{\text{ind}}\boldsymbol{x}_i^{t-dt})}{dt} + \frac{(\boldsymbol{\gamma}_i^t - \boldsymbol{\gamma}_i^{t-dt})\mathbf{\Delta}_m^{(b+1)\Delta t}}{dt}, \quad (15)$$

Then, we also have:

$$\boldsymbol{f_s}_i^t = {}^{\text{ind}}\boldsymbol{f_s}_i^t + \boldsymbol{\gamma_s}_i^t \mathbf{\Delta}_m^{(b+1)\Delta t}. \quad (16)$$

Indeed, from (2), we have :

$$\begin{aligned}\boldsymbol{f_s}_i^t &= \sum_{j\in\boldsymbol{\nu}_i} \boldsymbol{f_s}_{ij}^t = \sum_{j\in\boldsymbol{\nu}_i} \mathbf{K}_{ij}(\|\boldsymbol{d}_{ij}\| - l_{ij}^0)\frac{\boldsymbol{d}_{ij}}{\|\boldsymbol{d}_{ij}\|} \quad (17)\\ &= \sum_{j\in\boldsymbol{\nu}_i} \boldsymbol{f_s}_{ij}^t({}^{\text{ind}}\boldsymbol{x}_i^t + \boldsymbol{\gamma}_i^t\mathbf{\Delta}_m^{(b+1)\Delta t}, {}^{\text{ind}}\boldsymbol{x}_j^t + \boldsymbol{\gamma}_j^t\mathbf{\Delta}_m^{(b+1)\Delta t})\end{aligned}$$

Using a first order approximation of $\boldsymbol{f_s}_{ij}^t$, and using $\frac{\partial \boldsymbol{f_s}_{ij}}{\partial \boldsymbol{x}_j} = \frac{\partial \boldsymbol{f_s}_{ij}}{\partial \boldsymbol{d}_{ij}}\frac{\partial \boldsymbol{d}_{ij}}{\boldsymbol{x}_j} = -\frac{\partial \boldsymbol{f_s}_{ij}}{\partial \boldsymbol{x}_i}$, (17) can be rewritten as:

$$\boldsymbol{f_s}_i^t = \sum_{j\in\boldsymbol{\nu}_i} {}^{\text{ind}}\boldsymbol{f_s}_{ij}^t + \sum_{j\in\boldsymbol{\nu}_i} \frac{\partial \boldsymbol{f_s}_{ij}}{\partial \boldsymbol{x}_j}(\boldsymbol{\gamma}_j^t - \boldsymbol{\gamma}_i^t)\mathbf{\Delta}_m^{(b+1)\Delta t} \quad (18)$$

By comparing (18) to (16), we set: ${}^{\text{ind}}\boldsymbol{f_s}_i^t = \sum_{j\in\boldsymbol{\nu}_i} {}^{\text{ind}}\boldsymbol{f_s}_{ij}^t$, and $\boldsymbol{\gamma_s}_i^t = \sum_{j\in\boldsymbol{\nu}_i} \frac{\partial \boldsymbol{f_s}_{ij}}{\partial \boldsymbol{x}_j}(\boldsymbol{\gamma}_j^t - \boldsymbol{\gamma}_i^t)$. Furthermore, from (13), we have $\boldsymbol{\gamma}_m = \mathbf{I}_{3\text{x}3}$.

Returning to our hypothesis represented by (14), we obtain by updating the model using (5) and by the intermediate of (14) and (16):

$${\boldsymbol{x}_i}^{t+dt} = {}^{\text{ind}}{\boldsymbol{x}_i}^{t+dt} + \boldsymbol{\gamma}_i^{t+dt}\mathbf{\Delta}_m^{(b+1)\Delta t} \quad (19)$$

with:

$${}^{\text{ind}}{\boldsymbol{x}_i}^{t+dt} = {}^{\text{ind}}{\boldsymbol{x}_i}^t + \frac{dt^2}{m_i}{}^{\text{ind}}\boldsymbol{f_s}_i^t + (dt - \frac{dt^2}{m_i}D_v)^{\text{ind}}\dot{\boldsymbol{x}}_i^t,$$

$$\boldsymbol{\gamma}_i^{t+dt} = \boldsymbol{\gamma}_i^t + \frac{dt^2}{m_i}\boldsymbol{\gamma_s}_i^t + (1 - \frac{dt}{m_i}D_v)(\boldsymbol{\gamma}_i^t - \boldsymbol{\gamma}_i^{t-dt})$$

Eq. (19) thus validates our hypothesis concerning the decoupling. Furthermore, for all fixed nodes, we have $\boldsymbol{\gamma}_i = 0$.

Finally, by integrating (19) between $b\Delta t$ and $(b+1)\Delta t$ with a step time dt, we obtain $\frac{\Delta t}{dt}$ equations. By summing these equations we obtain:

$${\boldsymbol{x}_i}^{(b+1)\Delta t} = {\boldsymbol{x}_i}^{b\Delta t} + \boldsymbol{\gamma}_i^{(b+1)\Delta t}\mathbf{\Delta}_m^{(b+1)\Delta t} + {}^{\text{ind}}\boldsymbol{r}_i{}^{(b+1)\Delta t} \quad (20)$$

with:

$$\boldsymbol{\gamma}_i^{(b+1)\Delta t} = \sum_{t'_i}^{(b+1)\Delta t - dt} \frac{dt^2}{m_i}\boldsymbol{\gamma_s}_i^t + (1 - \frac{dt}{m_i}D_v)\boldsymbol{\gamma}_i^{((b+1)\Delta t - dt)} \quad (21)$$

$${}^{\text{ind}}\boldsymbol{r}_i{}^{(b+1)\Delta t} = \sum_{t=b\Delta t}^{(b+1)\Delta t - dt} (\frac{dt^2}{m_i}{}^{\text{ind}}\boldsymbol{f_s}_i^t + (dt - \frac{dt^2}{m_i}D_v)^{\text{ind}}\dot{\boldsymbol{x}}_i^t) \quad (22)$$

REFERENCES

[1] T. Wada, S. Hirai, and S. Kawamura, "Indirect simultaneous positioning operations of extensionally deformable objects," in *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, vol. 2, 1998, pp. 1333–1338.

[2] T. Wada, S. Hirai, S. Kawamura, and N. Kamiji, "Robust manipulation of deformable objects by a simple pid feedback," in *IEEE Int. Conf. on Robotics and Automation (ICRA)*, 2001, pp. 85–90.

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

[4] S. Kinio and A. Patriciu, "A comparative study of h∞ and pid control for indirect deformable object manipulation," in *IEEE Int. Conf. on Robotics and Biomimetics (ROBIO)*. IEEE, 2012, pp. 414–420.

[5] D. Navarro-Alarcon, H. M. Yip, Z. Wang, Y.-H. Liu, F. Zhong, T. Zhang, and P. Li, "Automatic 3-d manipulation of soft objects by robotic arms with an adaptive deformation model," *IEEE Trans. on Robotics*, vol. 32, no. 2, pp. 429–441, 2016.

[6] R. Lageau, A. Krupa, and M. Marchal, "Automatic Shape Control of Deformable Wires based on Model-Free Visual Servoing," *IEEE Robotics and Automation Letters*, vol. 5, no. 4, pp. 5252–5259, October 2020.

[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] C. A. Felippa, "A systematic approach to the element-independent corotational dynamics of finite elements," *Report CU-CAS-00-03, Center for Aerospace Structures, Colorado*, 2000.

[9] B. Lloyd, G. Székely, and M. Harders, "Identification of spring parameters for deformable object simulation," *IEEE Trans. on Visualization and Computer Graphics*, vol. 13, pp. 1081–94, October 2007.

[10] 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.

[11] M. L. Michelsen, "Application of semi-implicit runge—kutta methods for integration of ordinary and partial differential equations," *The Chemical Engineering Journal*, vol. 14, no. 2, pp. 107–112, 1977.

[12] Y. Bhasin and A. Liu, "Bounds for damping that guarantee stability in mass-spring systems," *Studies in health technology and informatics*, vol. 119, pp. 55–60, February 2006.

[13] Y. Duan, W. Huang, H. Chang, W. Chen, J. Zhou, S.-K. Teo, Y. Su, C.-K. Chui, and S. Chang, "Volume preserved mass-spring model with novel constraints for soft tissue deformation," *IEEE Journal of Biomedical and Health Informatics*, vol. 20, no. 1, pp. 268 – 280, 2016.

[14] E. Marchand, F. Spindler, and F. Chaumette, "Visp for visual servoing: a generic software platform with a wide class of robot control skills," *IEEE Robotics and Automation Magazine*, vol. 12, no. 4, pp. 40–52, 2005.

[15] C. B. Barber, D. P. Dobkin, and H. Huhdanpaa, "The quickhull algorithm for convex hulls," *ACM Transactions on Mathematical Software*, vol. 22, no. 4, p. 469–483, December 1996.

[16] C. Geuzaine and J.-F. Remacle, "Gmsh: A 3-d finite element mesh generator with built-in pre- and post-processing facilities," *Int. Journal for Numerical Methods in Engineering*, vol. 79, no. 11, pp. 1309–1331, 2009.

[17] A. Sobral and T. Bouwmans, "Bgs library: A library framework for algorithm's evaluation in foreground/background segmentation," 2014.