Optimal Shape Servoing with Task-focused Convergence Constraints

Victor H. Giraud, Maxime Padrin, Mohammadreza Shetab-Bushehri
Chedli Bouzgarrou, Youcef Mezouar, Erol Ozgur[1]

***Abstract*— Most deformable object manipulation tasks still rely on skillful human operators. To automate such tasks, a robotic system should not only be able to deform an object to a desired shape but also servo its deformation along a specific path towards the desired shape. We propose a shape servoing control scheme to automate such tasks. Our scheme controls the deformation trajectory towards the desired shape by imposing task-focused convergence constraints. The constraints impose how fast the different regions of the object converge to the desired shape. Integrating such a behavior in shape servoing forms our main contribution. Experiments, carried out on rubber layer assembly tasks, show that our control scheme outperforms a state-of-the-art shape servoing scheme.**

I. Introduction

The machines and robots have succeeded to automate many tedious tasks. However, there are still some that need to be automated effectively. Deformable object manipulation is one of them. It has a wide range of applications, for instance, in service robotics, surgical assistance [1], food packaging [2] and so on. The main challenge in deformable object manipulation is to generate the appropriate commands to control the object's shape. This is an open issue and it is known as *shape servoing* problem [3], [4], [5]. Although there are recent shape servoing schemes [6], [7], [8], they are not able to satisfy convergence constraints on the way towards the desired shape. The way to converge to the final shape is particularly critical to accomplish fine assembly tasks, such as the assembly of adhesive deformable surfaces.

We integrate shape convergence constraints into existing control schemes to obtain task-focused schemes. From a practical point of view, this should improve the manner of producing the desired outputs for the robotized tasks. In order to accomplish a shape servoing with convergence constraints, one should address the following challenges: (`C1`) deformable object perception, (`C2`) deformation prediction, (`C3`) under-actuated system control, and (`C4`) integration of convergence constraints within servoing scheme.

`C1`, `C2` are the primary challenges that each shape servoing scheme should address.

Therefore, we borrow existing solutions for them from a recent shape servoing scheme [8]. Thus, we focus only on challenge `C4`. In order to solve `C4`, we apply optimal control theory to encode implicitly the task-focused deformation behavior on a shape servoing scheme. We use neither trajectory nor subgoals. This will allow the robots to imitate the gestures of a skillful human operator on a dedicated task. We have named our scheme Optimal Shape Servoing (`OSS`).

`OSS` scheme contributes to the state-of-the-art as follows. *(i)* It allows encoding the task-focused deformation behaviour directly into a shape servoing scheme without any trajectory planning. The encoded behaviour imposes how fast the different regions of the object converge to the desired deformed shape. *(ii)* It is general in terms of the manipulated object topology, the shape perception method, and the task.

[1] Authors are with Université Clermont Auvergne, Clermont Auvergne INP, CNRS, Institut Pascal, 63000 Clermont-Ferrand, France. Email: {victor.giraud, maxime.padrin, sayed_mohammadreza.shetab_bushehri, chedli.bouzgarrou, youcef.mezouar, erol.ozgur}@sigma-clermont.fr

II. Related Work

A. Character Animation in Computer Graphics

Animating a deformable character/object from a given initial frame to a target frame under shape and motion constraints (*i.e.*, convergence constraints) is a well studied problem in computer graphics. We are indeed inspired from works such as [9] and [10] for our scheme.

B. Shape Servoing in Robotics

One of the first works in deformable object manipulation is the scheme described in [11]. It computed the deformation Jacobian of an object using the diminishing rigidity principle. It tested the method in simulations. It did not take into account any convergence constraints. [12] manipulated a thin-shell object. It modelled the object's deformation using FEM. It observed the object using a stereo camera. It controlled two points of the object, which were LEDs tracked by the stereo camera, in the plane. It used a closed-loop servoing scheme with an H^∞ controller. It did not take into account any convergence constraints. However, it included loop shaping constraints. It did not prove the control law stability.

[3] manipulated a volumetric object. It observed the object using a monocular RGB camera. It extracted the Fourier coefficients from the object's contour. It controlled the contour using these Fourier coefficients in the image. It used a proportional visual-servoing controller [13] with a Jacobian estimated online. It did not take into account any convergence constraints. It did not prove the control law stability.

[7] manipulated thin-shell and volumetric objects. It predicted the object's deformation using a model-free Jacobian estimated online. It observed a point cloud of a region of interest of the object using an RGB-D camera, and controlled it in the 3D space. It used a closed-loop scheme with a proportional gradient descent controller. It also used a sliding window weighted with a forgetting criteria. It did not take into account any convergence constraints. It proved the control law stability.

TABLE I

STATE-OF-THE-ART COMPARISON

	Object	**Vision Sensor**	**Servoing feature**	**Control space**	**Servoing scheme**	**Task-focused predefined trajectory**	**Task-focused convergence constraints**
[12], 2012	Thin-shell	Stereo	2 points	Plane	H^∞ control	No	No
[3], 2018	Volumetric	RGB	Contour	Image	Proportional control	No	No
[7], 2020	Volumetric	RGB-D	ROI point cloud	3D	Proportional control	No	No
[14], 2021	Volumetric	No sensor	Whole shape	3D	Open loop	Yes	No
[6], 2021	Volumetric	RGB	3 points	Image	Adaptive control	Yes	No
[8], 2022	Thin-shell	RGB	Whole shape	3D	Proportional control	No	No
OSS	Thin-shell (Volumetric or Linear)	RGB (RGB-D or Stereo)	Whole shape	3D	Optimal control	No	Yes

[14] manipulated a volumetric object. It modelled the object's deformation using Newton's method with Differential Dynamic Programming (DDP). It modelled the object's geometry with a tetrahedral mesh. It did not use any vision sensor, as its scheme was open-loop. It controlled the whole shape of the object in the 3D space. It did not take into account any convergence constraints but followed a trajectory generated through the object's deformation model.

[6] manipulated a linear, thin-shell, or volumetric object. It predicted the object's deformation with a Jacobian estimated online using Function Approximation Technique [15]. It observed the object using a monocular RGB camera. It controlled three points of the object in the image. It used a closed-loop scheme with an adaptive controller. It did not take into account any convergence constraints but followed a predetermined one. It included virtual force constraints to improve manipulability. It proved the control law stability.

[8] manipulated a thin-shell object. It modelled the object's deformation using As-Rigid-As-Possible (ARAP) method [16]. It observed the object's shape with a Shape-from-Template (SfT) algorithm [17] using a monocular RGB camera. It controlled the whole shape of the object in the 3D space. It used a closed-loop scheme with a proportional controller. It did not take into account any convergence constraints. It did not prove the control law stability as the scheme was strongly under-actuated.

C. Positioning OSS Scheme Compared to State-of-the-Art

To the best of authors' knowlegde, OSS is the only shape servoing scheme that does not require trajectory planning to achieve a task requiring specific intermediate shapes. It does so using the task-focused convergence constraints designed only once. While there exist schemes which followed task-focused predefined trajectories [14], [6] to achieve a task, they require to replan the trajectory each time the initial configuration changes. The OSS scheme is general. First, it can manipulate linear, thin-shell, or volumetric objects once a deformation Jacobian is available. Second, it can use any perception method providing 3D shape of the object, regardless of the sensor type. Third, the convergence constraints can be designed for a given specific task. Table I provides a comparison of the state-of-the-art schemes and the OSS scheme.

III. PROBLEM FORMULATION

A. Optimal Control over a Time Horizon

We represent the object's shape as a mesh grid made of nodes. Each node contains Cartesian coordinates of a vertex. Afterwards, we use a deformation Jacobian to predict the shape variations of the object as a function of the end-effectors' small displacements in quasi-static conditions. We then write the system's discrete time state equation as follows:

$$x(k+1) \;=\; x(k) + J(k)\,u(k)\,dt \tag{1}$$

where $k \in \mathbb{N}$ is the discretized time, $dt \in \mathbb{R}$ is the time step, $x \in \mathbb{R}^{3n}$ is the state vector formed by nodal points' coordinates, and n is the number of nodes in the object's mesh. $u \in \mathbb{R}^{6m}$ represents the control as the stacked vector of the velocity screws of the robots, where m is the number of robots. $J \in \mathbb{R}^{3n\times 6m}$ is the deformation Jacobian.

$$x = \begin{pmatrix} x_1 \\ y_1 \\ z_1 \\ x_2 \\ y_2 \\ z_2 \\ \vdots \end{pmatrix}_{3n\times 1} \quad \text{and} \quad u = \begin{pmatrix} v_{x_1} \\ v_{y_1} \\ v_{z_1} \\ w_{x_1} \\ w_{y_1} \\ w_{z_1} \\ \vdots \end{pmatrix}_{6m\times 1} \tag{2}$$

We define the shape error vector $e \in \mathbb{R}^{3n}$ as below:

$$e(k) = x(k) - x_d(k) \tag{3}$$

where $x_d \in \mathbb{R}^{3n}$ is the desired shape. Applying optimal control theory from [18], we write the global cost E in terms of the shape error cost and the control cost as follows:

$$E \;=\; \sum_{k=1}^{t_h} \big(\, e^\top(k)\, Q(k)\, e(k) + u^\top(k)\, R(k)\, u(k) \,\big) \tag{4}$$

where $t_h \in \mathbb{N}$ is the time horizon of our control. $Q(k) \in \mathbb{R}^{3n\times 3n}$ and $R(k) \in \mathbb{R}^{6m\times 6m}$ weigh the cost of the shape error and the control, respectively. The control u minimising E is written as:

$$u(k) = -K(k)\, e(k) \tag{5}$$

with $K(k) \in \mathbb{R}^{6m \times 3n}$ being defined as:

$$K(k) = [G^\top(k)\, S(k+1)\, G(k) + R(k+1)]^{-1}\, G^\top(k)\, S(k+1) \tag{6}$$

where $G(k) = dt\, J(k)$ and $S(k) \in \mathbb{R}^{3n \times 3n}$ is as follows:

$$\begin{cases} S(k) = S(k+1)\,[1 - G(k)\, K(k)] + Q(k) \\ S(t_h) = Q(t_h) \end{cases} \tag{7}$$

Figure 1 presents the block diagram of OSS scheme.

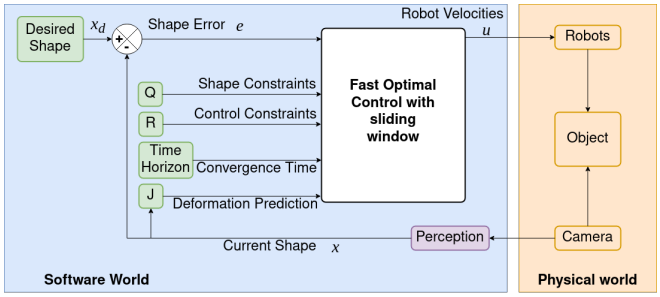

Fig. 1. The OSS scheme. Perception uses SfT.

Q matrix for shape constraints: The goal of Q matrix is not to establish reaching the intermediate shapes but how fast different regions of the object converge to the desired final shape. It can be constant or time varying. Its task-focused design remains a challenge and forms an open issue for future work. In the experimental section, we set it as a constant diagonal matrix based on the task descriptions. In the future, we plan to learn automatically Q from the observations of a human operator performing an industrial task.

R matrix for control constraints: It can prioritize some rotational or translational or both degrees of freedoms (dof) of the end-effectors of the robots over the other ones on a specific task. For instance, one can prioritize translational dofs on a cloth-like material shape servoing since the deformations are dominated mostly by the gravity and the effects of rotational dofs would remain relatively local. The R matrix can be constant or time varying.

Time horizon for task convergence time: The global cost E in (4) which includes the shape error is minimized over the time horizon. Thus the time horizon represents how fast the task will be finished (*i.e.*, settling time). Imposing a desired settling time for a complex task such as the shape servoing is not trivial with conventional control schemes (*e.g.*, PID). From a industrial point of view, this is a very useful property.

Jacobian J for deformation prediction: The deformation jacobian J depends on a deformation model representing the object's behaviour. It predicts how the object's shape will evolve under robots' motion. It can be computed either analytically or estimated on-line.

B. Algorithm

OSS scheme is detailed in algorithm 1. Line 1 starts the shape servoing control loop to be finished up to the time horizon. Line 2 perceives the object's current shape. Line 3 computes the shape error. Line 4 computes only the Jacobian

Algorithm 1 OSS scheme

Inputs:
x_d // Desired shape
dt // Time step (seconds)
t_h // Time horizon (number of iterations)
sw // Sliding window (number of iterations)
Q // Shape constraints (matrix or matrices)
R // Control constraints (matrix or matrices)

1: **for** k=1:t_h **do** // control loop over time horizon
2: x = PerceiveShape(object)
3: e = x - x_d
4: J = ComputeDeformationJacobian(x)
5: $G = dt\, J$
6: r_h = min(sw, t_h - k) // remaining horizon
7: S = $Q(k + r_h)$
8: **for** i=1:r_h **do** // recursive loop over sliding window
9: R_i = $R(k + r_h$ - i + 1)
10: Q_i = $Q(k + r_h$ - i)
11: $S = S - (SG(G^\top SG + R_i)^{-1})G^\top S + Q_i$
12: **end for**
13: $K = (G^\top SG + R)^{-1}G^\top S$
14: $u = -K\, e$ // control law
15: MoveRobots(u)
16: **end for**

of the current shape. It does not compute the Jacobians of the future shapes. Line 5 scales the Jacobian to form G matrix. Line 6 sets the remaining horizon as the minimum between the sliding window value and the remaining number of iterations up the time horizon. Line 7 initializes S for the recursive loop. Line 8 starts the recursive loop on the sliding window for the computation of S. S is thus computed over a shorter window rather than the longer time horizon. The Jacobian is constant during the recursive loop. This alleviates the computational cost. The recursive loop is also written considering a possible time varying Q and R matrices. Line 9 sets current iteration's R matrix. Line 10 sets current iteration's Q matrix. Line 11 computes recursively S. Line 12 ends the recursive loop. Line 13 computes the K matrix. Line 14 computes the control law of the robots. Line 15 applies the control law to the robots. Line 16 ends the shape servoing control loop.

IV. Experiments and Results

We validated the OSS scheme with a simulated task and one real-life task. The simulated task is shape-servoing of a translated thin-shell mesh. It is used for convergence time analysis. The real-life task is a rubber layer assembly. In this task, we place a piece of rubber in prolongation of another one that is fixed over a cylinder. In each task, we compare OSS scheme with ARAP-Shape-Servoing (ARAP-SS) [8].

A. Simulated Task for Convergence Time Analysis

1) Simulated case: In this task, we want to prioritize convergence of one side over the other, and of the center over the sides. The heatmap in figure 2 illustrates the gains

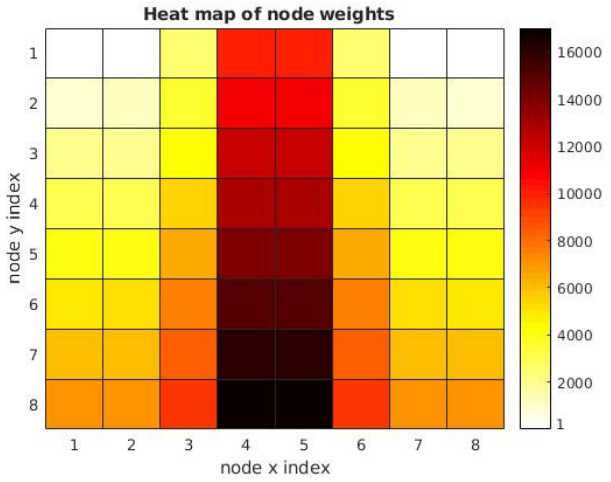

Fig. 2. Heatmap of the Q matrix values on the mesh nodes.

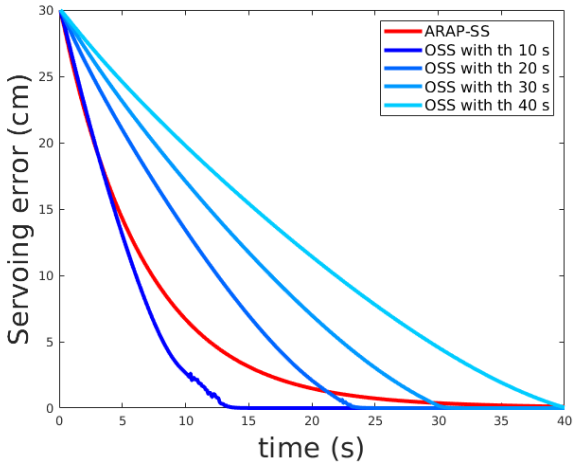

Fig. 4. Servoing errors versus time.

of our Q matrix on the mesh nodes. This matrix encodes the deformation behavior of the shape as seen in figure 3. We will focus in this section on the effect of the time horizon and sliding window parameters.

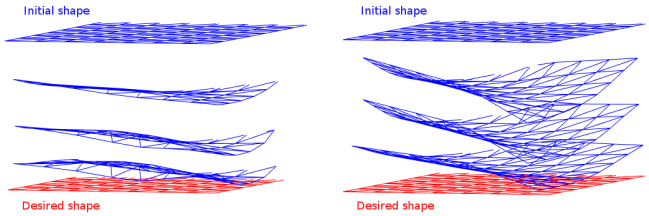

Fig. 3. ARAP-SS (left) and `OSS` schemes' convergence shapes (right). `OSS` scheme uses an encoded deformation behaviour.

2) Setup: We used an 8×8 mesh. It is deformed by two end-effectors grasping from the sides and moving freely. The control frequency is set as $10\,Hz$. Thus, the time step dt is $0.1\ s$. Consequently, a time horizon with 100 iterations would correspond to 10 seconds. We set the desired shape as a flat mesh. The initial shape is the same flat mesh translated $30\,cm$ upward. We used ARAP to model the object and to calculate the Jacobian. The R matrix is set as 0.001 times an identity matrix. This choice allows maximum freedom for the control.

3) Time horizon for task convergence time: We vary the time horizon parameter from $10\,s$ to $40\,s$ in order to analyse the task convergence times. We set the sliding window sizes equal to the time horizons (*e.g.*, $t_h = sw = 10s$, $t_h = sw = 20s$, and so on). The simulated results are shown in figure 4. One can observe that the task convergence times imposed by the time horizons are almost satisfied.

4) The impact of sliding window size: We recall that smaller the sliding window size, faster the computation of the control law. Therefore, we vary the sizes of the sliding windows from $t_h/2$ to $t_h/6$ in order to analyse the relation of the sliding window size to the task convergence time. We will look for the best compromise between the computational cost of the control law and the insurance of the task convergence time. Figure 5 shows the simulated results to reveal the relation between the sliding window sizes over the different time horizons and the convergence behaviours. A ratio of $t_h/4$ yields a good compromise between the computational cost and the insurance of the task convergence time.

B. Rubber Layer Assembly Task

1) Industrial use case: The rubber layer is rolled around a cylindrical support. The human operator aligns a mobile end to a fixed one as shown in figure 6. He follows two steps: he starts by deposing the back of the layer on the roller; then, he deposes the front of the layer by starting with the middle while sprawling progressively the lateral edges to match the fixed edge. Our objective is to automate this manual task.

2) Setup: We represent the mobile piece of rubber with a 7×10 mesh. To automate this task, we used two UR10 robot arms and an RGB-D camera as shown in figure 8. We implemented the shape perception (SfT) and shape deformation modeling (ARAP) algorithms presented in [8]. However our implementation runs on CPU while in [8] these were partly running in GPU. Our code uses ROS in order to reduce the transfer time from a lab experiment to industrial application. The code runs on a PC with Ubuntu 16.0.4 LTS, and Intel© Core™ i7-9850H CPU @ 2.60GHz processor. `OSS` scheme's implementation runs at $5Hz$ on this PC.

In order to obtain the desired shape seen in figure 7, we first aligned manually the mobile piece of rubber with the fixed piece on a cylinder, and then measured the shape with SfT. Although the camera is RGB-D, SfT uses only RGB.

3) Experiments: We analyzed the behaviour of the shape servoing schemes for different initial configurations. We tested 3 scenarios. In the first scenario, the initial configuration is a translated flat shape compared to the desired shape. In the second scenario the initial configuration is a translated convex shape. In the third scenario, the initial configuration is a translated concave shape.

To do so, we first formed a Q matrix with a very similar heatmap to one shown in figure 2 to be able to replicate the gestures of the human operator. In order to quantify the shape

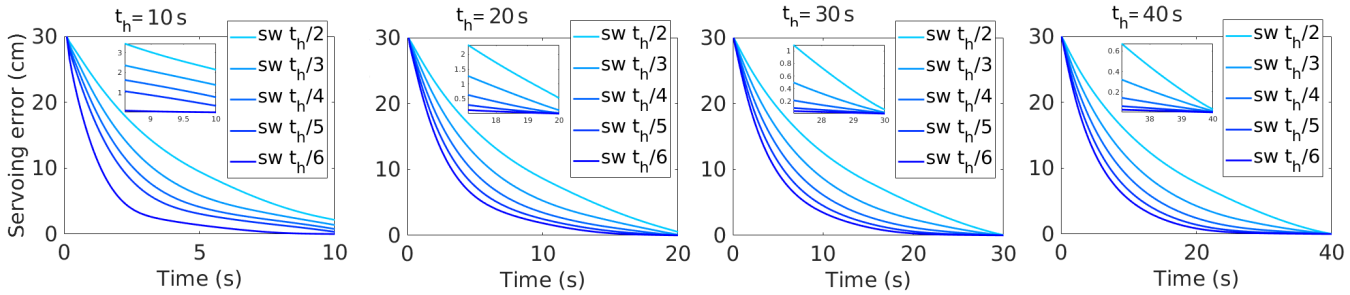

Fig. 5. OSS scheme servoing error versus time for the analysis of task convergence times. Small windows zoom the last few seconds.

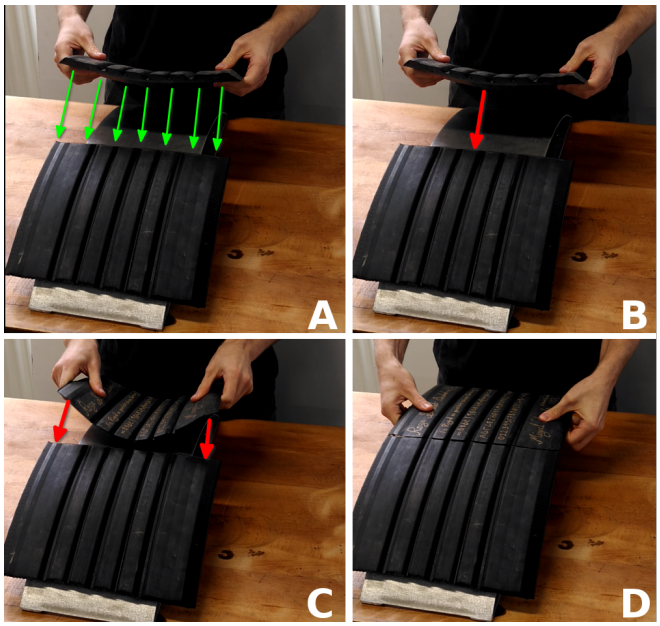

Fig. 6. Manual assembly of the rubber layer as it is done in the factory.

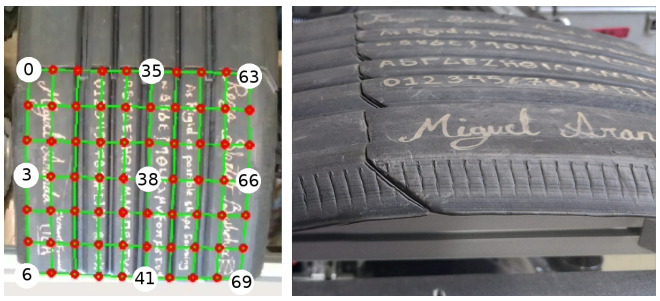

Fig. 7. Rubber layer with the perceived mesh in the desired configuration (left). The same desired configuration from a side view (right).

servoing, we then created reference trajectories of 9 nodes. These nodes are enumerated as 0, 3, 6, 35, 38, 41, 63, 66 and 69 in figure 7. Their trajectories are observed by SfT while performing manually the task from an initial configuration similar to the first scenario. The observed trajectories of the 9 nodes are then used as the implicit reference convergence behaviour for all the three scenarios regardless of different initial shapes. Figure 10, for instance, presents the results for the node number 35. One can observe that OSS scheme's node 35 performs a very similar behaviour while ARAP-SS' node 35 cannot.

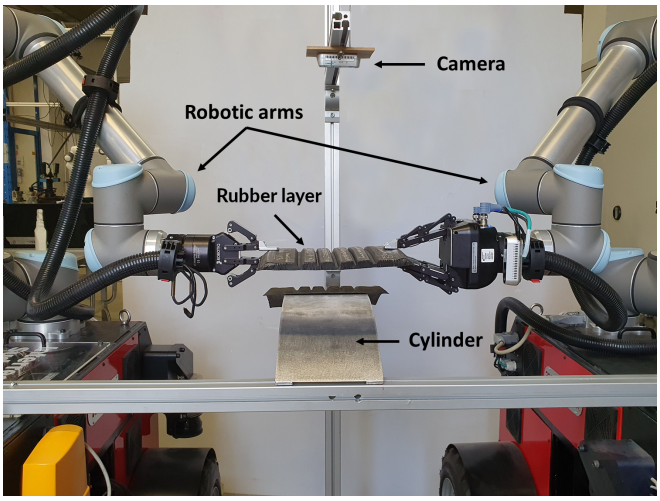

Fig. 8. Robotic setup for the rubber layer assembly task.

4) Results: We plotted the results of the 3 scenarios in figure 9. Each row presents the results of one scenario. The first column shows the initial configurations. The second column quantifies how well the nodes 0, 3 and 6 followed their respective implicit reference convergence behaviours. The curves represent the difference between the nodes' trajectories and their reference trajectories as explained above. Similarly, the second and third columns quantify how well the nodes 35, 38, 41 and 63, 66, 69 followed their respective implicit reference convergence behaviours. Again, the curves there represent the difference between the nodes' trajectories and their reference trajectories. The closer the curves to zero, the better the scheme's behaviour to replicate the manual task. One can observe that OSS scheme outperforms ARAP-SS in all scenarios. Although some spikes occurred on the curves because of noisy shape perception, the object's shape was stably servoed to the desired shape.

V. Conclusion

We presented a shape servoing scheme performing better than an existing state-of-the-art scheme on a real-life industrial task. Our scheme controls reaching deformation to a desired shape by imposing task-focused convergence constraints. Thus, it does not require trajectory planning even though the initial configuration changes.

As future work, we shall study (*i*) the shape servoing of different objects (linear, thin-shell, volumetric) on different

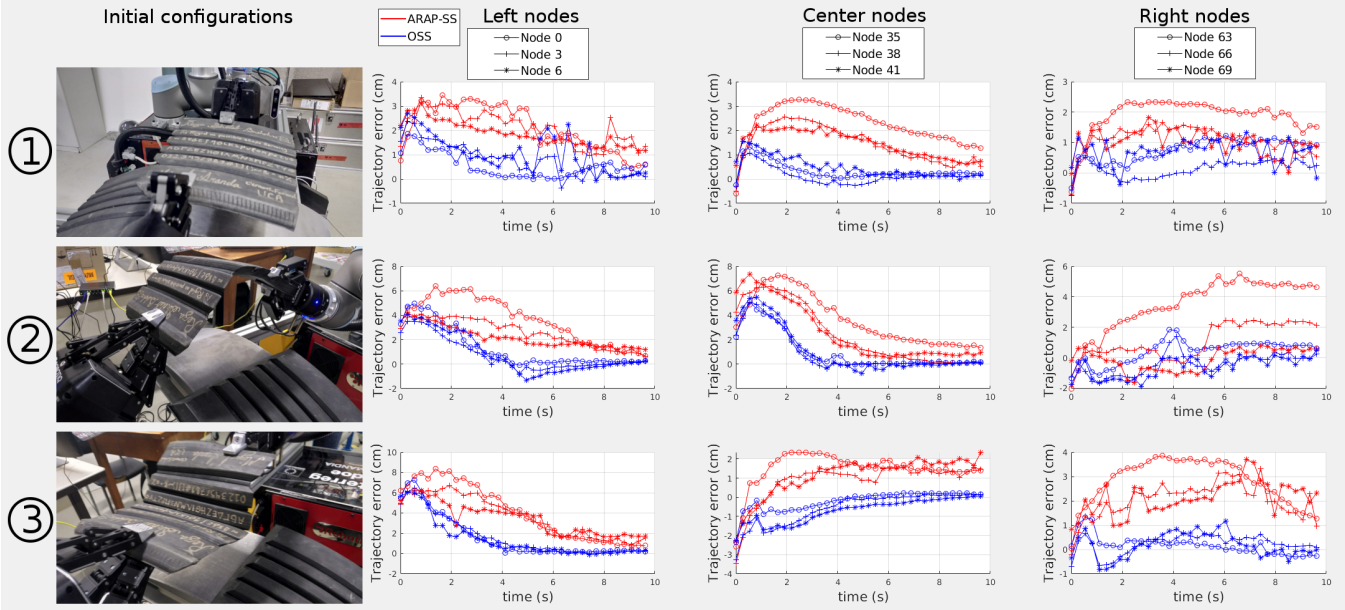

Fig. 9. Trajectory errors of the nodes in 3 scenarios with $t_h = 10\,s$, $sw = 2\,s$. Video of the experiment can be watched from here.

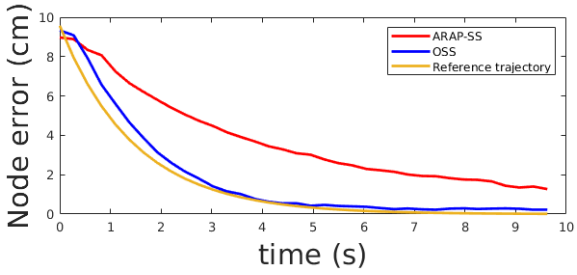

Fig. 10. The reference trajectory error of node 35 versus time compared with ARAP-SS and OSS schemes' node 35 behaviours.

tasks with specific constraints, (*ii*) the design methods of a Q matrix to be able encode a deformation behaviour, and (*iii*) the stability proof of the scheme.

ACKNOWLEDGEMENTS

This work is funded by SofTManBot project from the European Union's Horizon 2020 research and innovation program under grant agreement No 869855.

REFERENCES

[1] C. Shin, P. Fergusson, S. A. Pedram, J. Ma, E. P. Dutson, J.Rosen, "Autonomous Tissue Manipulation via Surgical Robot Using Learning Based Model Predictive Control", *IEEE International Conference on Robotics and Automation (ICRA)*, 2019.

[2] M. C. Gemici, A. Saxena, 'Learning Haptic Representation for Manipulating Deformable Food Objects' *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2014.

[3] 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 Transactions on Robotics*, vol. 34, no. 1, pp. 272-279, 2018.

[4] J. Sanchez, J. A. C. Ramon, B-C Bouzgarrou, Y. Mezouar, "Robotic Manipulation and Sensing of Deformable Objects in Domestic and Industrial Applications: A Survey", *The International Journal of Robotics Research*, 2018.

[5] F. Nadon, A. J. Valencia, P. Payeur, "Multi-Modal Sensing and Robotic Manipulation of Non-Rigid Objects: A Survey", *Robotics*, 2018.

[6] L. Han, H. Wang, Z. Liu, W. Chen, X. Zhang, "Visual Tracking Control of Deformable Objects with a FAT-based Controller" *IEEE Transactions on Industrial Electronics*, 2021.

[7] R. Lagneau, A. Krupa, M. Marchal, "Active Deformation through Visual Servoing of Soft Objects", *IEEE International Conference on Robotics and Automation (ICRA)*, 2020.

[8] M. Shetab-Bushehri, M. Aranda, Y. Mezouar, E. Ozgur, "As-Rigid-As-Possible Shape Servoing", *IEEE Robotics and Automation Letters - International Conference on Robotics and Automation (RAL-ICRA)*, 2022.

[9] J. Barbic, M. da Silva, J. Popovic, "Deformable Object Animation Using Reduced Optimal Control", *SIGGRAPH09: Special Interest Group on Computer Graphics and Interactive Techniques Conference*, 2009.

[10] C. Schulz, C. von Tycowicz, H. P. Seidel & K. Hildebrandt, "Animating deformable objects using sparse spacetime constraints", *ACM Transactions on Graphics (TOG), 33(4), 1-10.*, 2014.

[11] D. Berenson, "Manipulation of Deformable Objects Without Modeling and Simulating Deformation", *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2013.

[12] S. Kinio, A. Patriciu, "A Comparative Study of H^∞ and PID Control for Indirect Deformable Object Manipulation", *IEEE International Conference on Robotics and Biomimetics*, 2012.

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

[14] S. Zimmermann, R. Poranne, S. Coros, "Dynamic manipulation of deformable objects with implicit integration", *IEEE Robotics and Automation Letters*, 2021.

[15] D. Ebeigbe, T. Nguyen, H. Richter, and D. Simon, "Robust regressor-free control of rigid robots using function approximations", *IEEE Trans. Control Syst. Technol., vol. 28, no. 4, pp. 1433-1446*, 2020.

[16] O. Sorkine and M. Alexa, "As-rigid-as-possible surface modeling", *Eurographics Symposium on Geometry Processing*, 2007, pp. 109-116.

[17] E. Ozgur and A. Bartoli, "Particle-SfT: a Provably-Convergent, Fast Shape-from-Template Algorithm", *International Journal of Computer Vision*, vol. 123, no. 2, pp. 184-205, 2017.

[18] B. Anderson, J. Moore, "Optimal Control Linear Quadratic methods", *Dover Publications, Inc.*, 2007.