Shape Control of Deformable Linear Objects with Offline and Online Learning of Local Linear Deformation Models

Mingrui Yu, Hanzhong Zhong, and Xiang Li

***Abstract*— Shape control of deformable linear objects (DLOs) is challenging, since it is hard to obtain the deformation models. Previous studies often approximate the models in purely offline or online ways. This abstract proposes a scheme for shape control of DLOs, where the unknown model is estimated with both offline and online learning, allowing for both accurate modeling via offline learning and further updating for new DLOs via online learning. The simulation and real-world experiments show that the proposed method can achieve DLO shape control better than previous works.**

I. Introduction

Deformable linear objects (DLOs) refer to deformable objects in one dimension [1]. In the field of DLO shape control, a key challenge is to obtain the exact models of DLOs, because they are hard to calculate theoretically and vary among DLOs [2]. Some analytical modeling methods can be used to model DLOs [3]–[5]. However, all are approximate models, and require accurate parameters of DLOs which are difficult to acquire. Data-driven approaches have been applied to learn the deformation models. A common method is to first offline learn a forward kinematics model (FKM) offline, and then use model predictive control (MPC) in manipulation [6]–[9]. Reinforcement learning and imitation learning methods have also been studied [10]–[13]. The problem is that they are less data-efficient and may get into trouble manipulating an untrained DLO. Apart from these offline approaches, some studies have used purely online methods to estimate the local linear deformation model of manipulated DLOs, which can be applied to any new DLO [14]–[17]. However, these online estimated models are less accurate because only limited local data can be utilized.

In this abstract, we propose a scheme for shape control of DLOs, where the unknown deformation model is estimated with both offline and online learning, shown in Fig. 1. Specifically, we use a radial-basis-function neural network (RBFN) to model the mapping from the state to the local linear deformation model. In the offline phase, the RBFN is trained on random data. The offline model is then seamlessly migrated to the online phase as an initial estimation. In the online phase, an adaptive controller is proposed to control the shape, in which the RBFN is further updated to adapt to the manipulated DLO concurrently. Thus, the offline learning and online learning complement each other. The system's stability is analyzed using the Lyapunov method. Simulation and real-world experiment results are presented to demonstrate the better performance of the proposed scheme compared with the previous methods. The full paper [18], video, and code are available at the project website[1]. A further improved work can be found here[2].

M. Yu, H. Zhong, and X. Li are with the Department of Automation, Tsinghua University, China. Corresponding author: Xiang Li (xiangli@tsinghua.edu.cn)

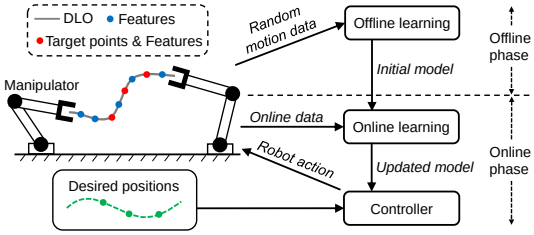


Fig. 1. Overview of the proposed scheme for DLO shape control. The shape of the DLO is represented by multiple features along the DLO. Some of the features are chosen as target points, and the task is defined as moving the target points to their desired positions.

II. Methodology

This abstract considers quasi-static shape control of elastic DLOs, as illustrated in Fig. 1. Some frequently-used notations are listed as follows. The vertical concatenation of column vector $\boldsymbol{a}$ and $\boldsymbol{b}$ is denoted as $[\boldsymbol{a};\boldsymbol{b}]$. The position vector of the end-effectors is represented as $\boldsymbol{r} \in \Re^n$. The position of the i^{th} feature is represented as $\boldsymbol{x}_i \in \Re^l$. The overall shape vector of the DLO is represented as $\boldsymbol{x} = [\boldsymbol{x}_1;\cdots;\boldsymbol{x}_m] \in \Re^{lm}$, where m is the number of the features.

A. Local Linear Deformation Model

The velocity vector of the DLO features can be locally linearly related to the velocity vector of the end-effectors using a Jacobian matrix [14]–[17]. Different from the previous works, we estimate the Jacobian matrix by learning the mapping from the state $(\boldsymbol{x},\boldsymbol{r})$ to the Jacobian matrix $\boldsymbol{J}$:

$$\dot{\boldsymbol{x}} = \boldsymbol{J}(\boldsymbol{x},\boldsymbol{r})\dot{\boldsymbol{r}} \tag{1}$$

Proposition: With the quasi-static assumption, the velocity vector of the features on the elastic DLO can be related to the velocity vector of the end-effectors as (1).

Proof: Denote the potential energy of the elastic DLO as E, which is assumed to be fully determined by $\boldsymbol{x}$ and $\boldsymbol{r}$. In the quasi-static assumption, internal equilibrium holds at all states during the manipulation. That is, $\partial E/\partial \boldsymbol{x} = \boldsymbol{0}$ at any state. Consider the DLO is moved from state $(\bar{\boldsymbol{x}},\bar{\boldsymbol{r}})$ to state $(\bar{\boldsymbol{x}}+\delta\boldsymbol{x},\bar{\boldsymbol{r}}+\delta\boldsymbol{r})$ where $\delta\boldsymbol{x}$ and $\delta\boldsymbol{r}$ are small displacements of the features and the end-effectors. Denote $\partial E/\partial \boldsymbol{x}$ as $\boldsymbol{g}(\boldsymbol{x},\boldsymbol{r})$,

[1]https://mingrui-yu.github.io/shape_control_DLO/
[2]https://mingrui-yu.github.io/shape_control_DLO_2/

$\partial^2 E/(\partial \boldsymbol{x}\partial \boldsymbol{x})$ as $\boldsymbol{A}(\boldsymbol{x},\boldsymbol{r})$, and $\partial^2 E/(\partial \boldsymbol{x}\partial \boldsymbol{r})$ as $\boldsymbol{B}(\boldsymbol{x},\boldsymbol{r})$. Using Taylor expansion and neglecting higher order terms:

$$\boldsymbol{g}(\bar{\boldsymbol{x}}+\delta\boldsymbol{x},\bar{\boldsymbol{r}}+\delta\boldsymbol{r}) \approx \boldsymbol{g}(\bar{\boldsymbol{x}},\bar{\boldsymbol{r}}) + \boldsymbol{A}(\bar{\boldsymbol{x}},\bar{\boldsymbol{r}})\delta\boldsymbol{x} + \boldsymbol{B}(\bar{\boldsymbol{x}},\bar{\boldsymbol{r}})\delta\boldsymbol{r} \quad (2)$$

where $\boldsymbol{g}(\bar{\boldsymbol{x}}+\delta\boldsymbol{x},\bar{\boldsymbol{r}}+\delta\boldsymbol{r}) = \boldsymbol{g}(\bar{\boldsymbol{x}},\bar{\boldsymbol{r}}) = \boldsymbol{0}$. Assuming the DLO has a positive and full-rank stiffness matrix around the equilibrium point, matrix $\boldsymbol{A}$ is invertible [14]. Then, we have

$$\delta\boldsymbol{x} \approx -\left(\boldsymbol{A}(\bar{\boldsymbol{x}},\bar{\boldsymbol{r}})\right)^{-1}\boldsymbol{B}(\bar{\boldsymbol{x}},\bar{\boldsymbol{r}})\delta\boldsymbol{r} \quad (3)$$

In slow manipulations, $\dot{\boldsymbol{x}} \approx \delta\boldsymbol{x}/\delta t$ and $\dot{\boldsymbol{r}} \approx \delta\boldsymbol{r}/\delta t$ with small δt. Then, denoting $-(\boldsymbol{A}(\boldsymbol{x},\boldsymbol{r}))^{-1}\boldsymbol{B}(\boldsymbol{x},\boldsymbol{r})$ as $\boldsymbol{J}(\boldsymbol{x},\boldsymbol{r})$, we derive (1) and prove the proposition.

Note that (1) can be rewritten as

$$\dot{\boldsymbol{x}} = \begin{bmatrix} \dot{\boldsymbol{x}}_1 \\ \vdots \\ \dot{\boldsymbol{x}}_m \end{bmatrix} = \begin{bmatrix} \boldsymbol{J}_1(\boldsymbol{x},\boldsymbol{r}) \\ \vdots \\ \boldsymbol{J}_m(\boldsymbol{x},\boldsymbol{r}) \end{bmatrix} \dot{\boldsymbol{r}} \quad (4)$$

where $\boldsymbol{J}_k(\boldsymbol{x},\boldsymbol{r})$ is the $((k-1)l+1)^{\text{th}}$ to $(kl)^{\text{th}}$ rows of $\boldsymbol{J}(\boldsymbol{x},\boldsymbol{r})$. Thus, it can be obtained that

$$\dot{\boldsymbol{x}}_k = \boldsymbol{J}_k(\boldsymbol{x},\boldsymbol{r})\dot{\boldsymbol{r}}, \quad k = 1,\cdots,m \quad (5)$$

which indicates that different features correspond to different Jacobian matrices.

B. Offline Learning

We apply a neural network (NN) to approximate the Jacobian matrix, in which the input is the current state and the output is the Jacobian. Two properties of the Jacobian can be noticed intuitively: 1) translation-invariance: translation of the whole DLO will not alter the Jacobian matrix; 2) approximate scale-invariance: DLOs with different lengths but similar overall shapes may have similar Jacobian matrices. Thus, to improve the NN's generalization ability, we modify the representation of the input state from $[\boldsymbol{x};\boldsymbol{r}]$ to

$$\boldsymbol{\phi} \triangleq [\bar{\boldsymbol{x}}_1;\cdots;\bar{\boldsymbol{x}}_m;\ \bar{\boldsymbol{p}};\boldsymbol{q}_0;\boldsymbol{q}_1] \quad (6)$$

where

$$\bar{\boldsymbol{x}}_k = \left[\frac{\boldsymbol{x}_k-\boldsymbol{p}_0}{\|\boldsymbol{x}_k-\boldsymbol{p}_0\|};\frac{\boldsymbol{x}_k-\boldsymbol{p}_1}{\|\boldsymbol{x}_k-\boldsymbol{p}_1\|}\right], \quad \bar{\boldsymbol{p}} = \frac{\boldsymbol{p}_1-\boldsymbol{p}_0}{\|\boldsymbol{p}_1-\boldsymbol{p}_0\|} \quad (7)$$

$(k = 1,\cdots,m)$, where $\boldsymbol{p}_0$, $\boldsymbol{p}_1$ are the positions of the left and right grasped ends of the DLO, and $\boldsymbol{q}_0$, $\boldsymbol{q}_1$ are the orientations of the left and right grasped ends.

Then, (5) can be rewritten as

$$\dot{\boldsymbol{x}}_k = \boldsymbol{J}_k(\boldsymbol{\phi})\dot{\boldsymbol{r}}, \quad k = 1,\cdots,m \quad (8)$$

We apply a radial-basis-function network (RBFN) [19] to represent the actual Jacobian matrix as a function of $\boldsymbol{\phi}$:

$$\text{vec}\left(\boldsymbol{J}_k(\boldsymbol{\phi})\right) = \boldsymbol{W}_k\boldsymbol{\theta}(\boldsymbol{\phi}), \quad k = 1,\cdots,m \quad (9)$$

where $\text{vec}(\cdot)$ refers to the column vectorization operator, and $\boldsymbol{W}_k$ is unknown actual RBFN weights for the k^{th} feature. The $\boldsymbol{\theta}(\boldsymbol{\phi})$ represents the vector of activation functions. We use the gaussian radial function as the activation function. Equation (9) can be decomposed as

$$\boldsymbol{J}_{ki}(\boldsymbol{\phi}) = \boldsymbol{W}_{ki}\boldsymbol{\theta}(\boldsymbol{\phi}), \quad i = 1,\cdots,n \quad (10)$$

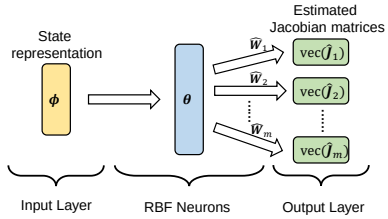


Fig. 2. The architecture of the RBFN for learning the local linear deformation model. The network takes the state representation in (6) as the input and outputs the estimated Jacobian matrices.

where $\boldsymbol{J}_{ki}$ is the i^{th} column of $\boldsymbol{J}_k$, and $\boldsymbol{W}_{ki}$ is the $((i-1)l+1)^{\text{th}}$ to $(il)^{\text{th}}$ rows of $\boldsymbol{W}_k$. Subscribing (10) into (8) yields

$$\dot{\boldsymbol{x}}_k = \boldsymbol{J}_k(\boldsymbol{\phi})\dot{\boldsymbol{r}} = \sum_{i=1}^{n}\boldsymbol{J}_{ki}(\boldsymbol{\phi})\dot{r}_i = \sum_{i=1}^{n}\boldsymbol{W}_{ki}\boldsymbol{\theta}(\boldsymbol{\phi})\dot{r}_i \quad (11)$$

where $\dot{r}_i$ is the i^{th} element of $\dot{\boldsymbol{r}}$. The estimated Jacobian matrix is represented as

$$\text{vec}(\hat{\boldsymbol{J}}_k(\boldsymbol{\phi})) = \hat{\boldsymbol{W}}_k\boldsymbol{\theta}(\boldsymbol{\phi}) \quad (12)$$

where $\hat{\boldsymbol{W}}$ is estimated weights. The architecture of the RBFN is shown in Fig. 2. The approximation error for the k^{th} feature $\boldsymbol{e}_k$ is specified as

$$\begin{aligned}\boldsymbol{e}_k &= \dot{\boldsymbol{x}}_k - \hat{\boldsymbol{J}}_k(\boldsymbol{\phi})\dot{\boldsymbol{r}} \\ &= \sum_{i=1}^{n}\boldsymbol{W}_{ki}\boldsymbol{\theta}(\boldsymbol{\phi})\dot{r}_i - \sum_{i=1}^{n}\hat{\boldsymbol{W}}_{ki}\boldsymbol{\theta}(\boldsymbol{\phi})\dot{r}_i = \sum_{i=1}^{n}\Delta\boldsymbol{W}_{ki}\boldsymbol{\theta}(\boldsymbol{\phi})\dot{r}_i\end{aligned} \quad (13)$$

We use the smooth $L1$ loss [20] of $\boldsymbol{e}_k$ for offline training.

C. Adaptive Control through Online Learning

We propose an adaptive control scheme, in which the offline estimated model is treated as an initial approximation and then further updated during the shape control tasks. The target points can be any subset of the features, whose indexes form set $\mathcal{C}$. Then, the target shape vector $\boldsymbol{x}^c$ and target Jacobian matrix $\boldsymbol{J}^c(\boldsymbol{\phi})$ are denoted as $\boldsymbol{x}^c = [\cdots;\boldsymbol{x}_k;\cdots], \boldsymbol{J}^c(\boldsymbol{\phi}) = [\cdots;\boldsymbol{J}_k(\boldsymbol{\phi});\cdots], k \in \mathcal{C}$. The velocity vector of the robot end-effectors $\dot{\boldsymbol{r}}$ is controlled and specified as

$$\dot{\boldsymbol{r}} = -\alpha\left(\hat{\boldsymbol{J}}^c(\boldsymbol{\phi})\right)^{\dagger}\Delta\boldsymbol{x}^c \quad (14)$$

where $\left(\hat{\boldsymbol{J}}^c(\boldsymbol{\phi})\right)^{\dagger}$ is the Moore-Penrose pseudo-inverse of the estimated Jacobian matrix. In addition, $\Delta\boldsymbol{x}^c = \boldsymbol{x}^c - \boldsymbol{x}^c_{desired}$ where $\boldsymbol{x}^c_{desired}$ is the desired position vector of the target points, and $\alpha \in \Re$ is a positive control gain.

The online updating law of the j^{th} row of $\hat{\boldsymbol{W}}_{ki}$ of the RBFN is specified as

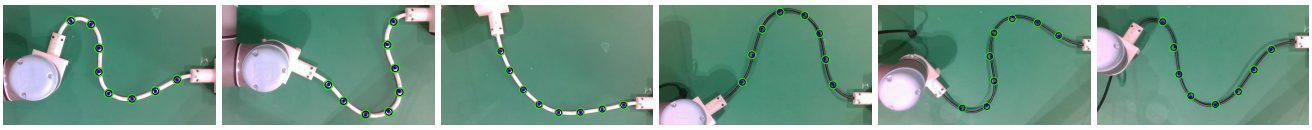
$$\dot{\hat{\boldsymbol{W}}}^T_{kij} = \dot{r}_i\boldsymbol{\theta}(\boldsymbol{\phi})(\eta_1\Delta x_{kj} + \eta_2 e_{kj}), \quad j = 1,\cdots,l \quad (15)$$

where Δx_{kj} is the j^{th} element of the task error $\Delta\boldsymbol{x}_k$, and e_{kj} is the j^{th} element of the approximation error $\boldsymbol{e}_k$. The η_1 and η_2 are positive scalars. Such updating is done for all $k \in \mathcal{C}$ and $i = 1,\cdots,n$.

TABLE I

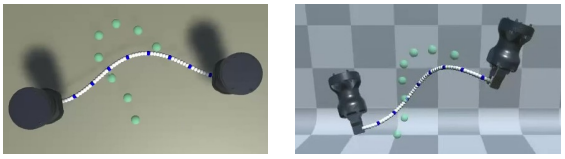
PERFORMANCE OF THE METHODS IN 2D AND 3D DLO SHAPE CONTROL TASKS IN SIMULATION.

Methods	2D tasks				3D tasks			
	Offline training samples	Success rate	Average task error (cm)	Average task time (s)	Offline training samples	Success rate	Average task error (cm)	Average task time (s)
FKM+MPC	180k	82/100	1.662	10.488	180k	52/100	3.298	14.275
WLS	-	85/100	0.992	14.351	-	55/100	1.940	20.214
SAC	1000k	42/100	3.185	6.486	1000k	10/100	3.412	8.070
Ours(w/o online)	**30k**	94/100	0.461	8.568	**30k**	69/100	1.446	9.521
Ours	**30k**	**97/100**	**0.457**	8.512	**30k**	**92/100**	**1.254**	9.529



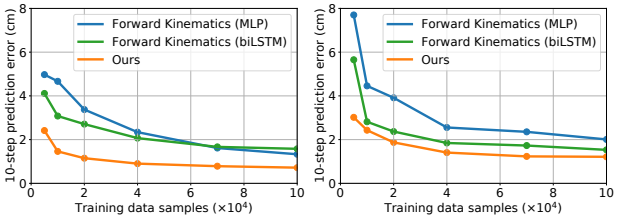
(a) Case 1 (b) Case 2 (c) Case 3 (d) Case 6 (e) Case 7 (f) Case 8

Fig. 3. Some of the shape control tasks accomplished using our method in real-world experiments. The left end of the DLO is grasped by a UR5 arm and the right end is fixed. The green+black circles represent the desired positions of the DLO features. In all cases, the DLO starts from a straight line.



(a) 2D (b) 3D

Fig. 4. The simulation environment. The blue points represent the features along the DLO. The green points represent the desired positions of the features.



(a) 2D (b) 3D

Fig. 5. The relationship between the offline modeling accuracy and the amount of training data.

III. RESULTS

We carry out both simulation and real-world experiments to validate the proposed method. The simulation environment is shown in Fig. 4. We choose three representative classes of methods for comparison: 1) learning FKM offline and using MPC for shape control (FKM+MPC); 2) estimating the Jacobian matrix online using weighted least square estimation (WLS); 3) reinforcement learning (SAC).

A. Offline Learning of the Deformation Model

We test the offline modeling accuracy on a certain DLO and its relationship to the amount of training data, in which we compare our local linear Jacobian model with nonlinear forward kinematics models based on multi-layer perceptrons (MLP) or biLSTM. The data are randomly collected in the simulation. As shown in Fig. 5, the results indicate that our Jacobian model can achieve higher prediction accuracy with less training data.

TABLE II

PERFORMANCE IN REAL-WORLD 2D SHAPE CONTROL TASKS.

Methods	Success rate	Average task error (cm)	Average task time (s)
FKM+MPC	**10/10**	1.394	7.670
WLS	9/10	1.164	19.089
SAC	1/10	4.955	12.300
Ours(w/o online)	**10/10**	1.153	10.090
Ours	**10/10**	**0.620**	8.220

B. Shape Control with Online Learning

We evaluate the proposed control method and compare it with other methods. If the final task error is less than 5cm, the task is regarded successful. The *average task error* refers to the average task error of only the successful cases.

1) Simulation: First, we test their performance in both 2D and 3D DLO shape control tasks in simulation. The manipulated DLO is a new DLO not used in the offline learning, and 100 cases with different feasible desired shapes are tested. As shown in Table I, our method significantly outperforms the compared methods on both success rate and average task error, even using much less offline training data.

2) Real-world experiments: We also evaluate these methods in real-world 2D tasks, as shown in Fig. 3. The same offline models as those in the simulation are used (no real-world data are collected offline). We separately carry out 5 tests with different feasible desired shapes on two different DLOs. The results are shown in Table II. Our method completes all 10 tasks and achieves the lowest average task error, where the online learning enables faster and more precise control.

IV. CONCLUSION

This abstract considers the shape control of DLOs with unknown deformation models. First, the offline learning well initiates the estimation of the model. Then, the adaptive control scheme with online learning further updates the model and achieves shape control. The experiments demonstrate the effectiveness of our method. For more details, please refer to the full paper [18].

REFERENCES

[1] J. Sanchez, J.-A. Corrales, B.-C. Bouzgarrou, and Y. Mezouar, "Robotic manipulation and sensing of deformable objects in domestic and industrial applications: a survey," *The International Journal of Robotics Research*, vol. 37, no. 7, pp. 688–716, 2018.

[2] J. Zhu, A. Cherubini, C. Dune, D. Navarro-Alarcon, F. Alambeigi, D. Berenson, F. Ficuciello, K. Harada, J. Kober, X. Li, *et al.*, "Challenges and outlook in robotic manipulation of deformable objects," *IEEE Robotics and Automation Magazine*, 2021.

[3] H. Yin, A. Varava, and D. Kragic, "Modeling, learning, perception, and control methods for deformable object manipulation," *Science Robotics*, vol. 6, no. 54, 2021.

[4] S. Duenser, J. M. Bern, R. Poranne, and S. Coros, "Interactive robotic manipulation of elastic objects," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2018, pp. 3476–3481.

[5] A. Koessler, N. Roca Filella, B. Bouzgarrou, L. Lequievre, and J.-A. Corrales Ramon, "An efficient approach to closed-loop shape control of deformable objects using finite element models," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*, 2021.

[6] W. Yan, A. Vangipuram, P. Abbeel, and L. Pinto, "Learning predictive representations for deformable objects using contrastive estimation," in *4th Conference on Robot Learning (CoRL)*, 2020.

[7] W. Zhang, K. Schmeckpeper, P. Chaudhari, and K. Daniilidis, "Deformable linear object prediction using locally linear latent dynamics," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*, 2021.

[8] M. Yan, Y. Zhu, N. Jin, and J. Bohg, "Self-supervised learning of state estimation for manipulating deformable linear objects," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 2372–2379, 2020.

[9] Y. Yang, J. A. Stork, and T. Stoyanov, "Learning to propagate interaction effects for modeling deformable linear objects dynamics," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*, 2021.

[10] X. Lin, Y. Wang, J. Olkin, and D. Held, "Softgym: Benchmarking deep reinforcement learning for deformable object manipulation," in *4th Conference on Robot Learning (CoRL)*, 2020.

[11] L. Rita and K. Yiannis, "Learning shape control of elastoplastic deformable linear objects," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*, 2021.

[12] A. Nair, D. Chen, P. Agrawal, P. Isola, P. Abbeel, J. Malik, and S. Levine, "Combining self-supervised learning and imitation for vision-based rope manipulation," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, 2017, pp. 2146–2153.

[13] T. Tang, C. Wang, and M. Tomizuka, "A framework for manipulating deformable linear objects by coherent point drift," *IEEE Robotics and Automation Letters*, vol. 3, no. 4, pp. 3426–3433, 2018.

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

[15] J. Zhu, B. Navarro, P. Fraisse, A. Crosnier, and A. Cherubini, "Dual-arm robotic manipulation of flexible cables," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2018, pp. 479–484.

[16] S. Jin, C. Wang, and M. Tomizuka, "Robust deformation model approximation for robotic cable manipulation," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2019, pp. 6586–6593.

[17] R. Lagneau, 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, 2020.

[18] M. Yu, H. Zhong, and X. Li, "Shape control of deformable linear objects with offline and online learning of local linear deformation models," in *2022 IEEE International Conference on Robotics and Automation (ICRA)*, 2022.

[19] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[20] R. Girshick, "Fast r-cnn," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 1440–1448.