DeformerNet: A Deep Learning Approach to 3D Deformable Object Manipulation

Bao Thach*, Alan Kuntz*, Tucker Hermans*†
*University of Utah Robotics Center; †NVIDIA

I. Introduction

Following Navarro-Alarcon *et al.* [5], we adopt the terminology *shape servo* to describe the manipulation task in which a robot manipulates a 3D deformable object into a desired shape. While performing this servo control, the robot estimates the state of the object with visual sensors and uses this as a feedback signal. However, the biggest question is: How do we obtain a good state representation of the object to inform the robot about the object's shape? *3D* in this context refers to *triparametric* or *volumetric* objects [12] which have no dimension significantly smaller than the other two, unlike *uniparametric* (rope) and *biparametric* objects (cloth).

A series of papers [6, 7, 8] define a set of *feature points* on the object as the state representation. These methods only work for known objects with distinct texture and cannot generalize to a diverse set of objects. This formulation also simply controls the displacements of individual points which does not fully reflect the 3D shape of the object. For precise control, one must use a large number of feature points, making control highly susceptible to noise and occlusion. Navarro-Alarcon and Liu [5] and Qi et al. [10] leverage 2D image contours as the feedback representations. However, using only 2D data severely limits the space of deformation control in 3D.

Hu et al. [1] address these shortcomings by using point clouds. They use extended FPFH [11] to extract a feature vector from the point cloud and learn the deformation function via a Deep Neural Network (DNN). We show that this hard-coded feature descriptor fails to generalize well.

Previous learning-based methods such as [4, 14, 15] focus on rope and cloth. These works, often operated in image space, are not quite relevant to 3D object manipulation problems. There are also physical differences between rope-cloth and 3D objects (e.g. a 3D elastic object like soft tissue will return to its initial shape after the robot releases it), which makes methods in one domain not applicable to the other.

Therefore, we propose a novel deep learning approach to solving the 3D shape servo problem. Instead of using the hard-coded feature vector as in [1], we develop a DNN that takes point clouds of the deformable objects as the inputs and outputs feature vectors. In addition, we develop a DNN that maps these feature vectors to the desired end-effector's Cartesian position. We train both the feature extraction and deformation control neural networks together in an end-to-end fashion. Once trained, given point clouds of the object's current and goal shapes, the robot computes the desired position of its gripper at every time step. Finally, we study the problem of predicting the manipulation point. We are the first to propose a solution to this problem for 3D shape servoing.

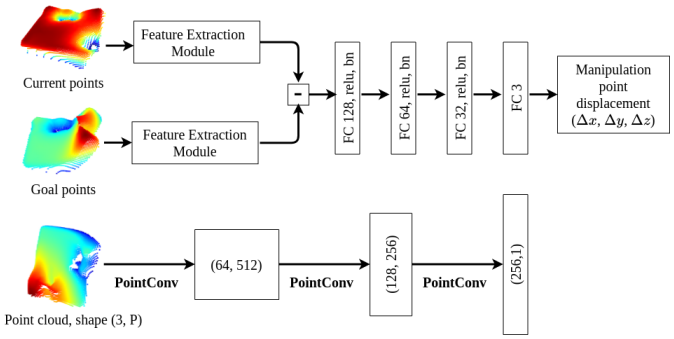


Fig. 1: (Top) Architecture of DeformerNet; (Bottom) architecture of the feature extraction module.

II. Shape Servoing

We define a set of manipulation points p_m located on the surface of the deformable object where the robot gripper makes contact with the object and can directly control their positions. We formulate the shape servoing problem as finding a deformation function F which maps the state representation ψ of the deformable object to the manipulation points' positions $p = [x, y, z] \in \mathbb{R}^3$. The intuition here is that we can drive the object to a desired shape by controlling the locations of these manipulation points: $p = F(\psi)$.

We choose ψ to be a feature vector which encodes the shape of the deformable object. We leverage a DNN derived from PointConv [13] which takes the point cloud of the current object shape and that of the goal object shape as the inputs and learns a 256-dimension feature vector. Figure 1 visualizes the network architecture of our feature extraction network.

For closed-loop operations, we modify the original equation to reflect the position and feature displacements: $\Delta p = F(\Delta\psi)$ where $\Delta\psi = \psi^* - \psi$ is the difference between the feature vector of the goal shape and that of the current object shape, and $\Delta p = p^* - p = [\Delta x, \Delta y, \Delta z]^T$ is the relative displacement between the desired and current manipulation points. Δp is also the Cartesian displacement of the robot end-effector since the robot directly controls the positions of the manipulation points.

We define this deformation function F as a fully-connected neural network. The goal of the shape servoing problem then becomes learning a model which maps the feature vector displacement $\Delta\psi$ to the desired displacement of the manipulation points Δp. Given the learned F, we can now compute the

desired end-effector position at every time step. The entire network architecture is shown in Fig. 1.

At runtime, the robot is given the point cloud of the current object shape and the point cloud of the goal object shape. First, we pass the point clouds through DeformerNet which outputs the desired gripper position. We then use the RRT-connect motion planner to plan a joint space path following the desired end-effector displacement. After the robot has reached an intermediate desired position, it gets a new point cloud of the current object shape and repeats the process.

Prior works [1, 6, 7, 8] assume that the manipulation points are given to the robot. However, a robot should select the best possible points to achieve the task. To understand the importance of selecting a good manipulation point, consider a simple scenario in Fig. 2. The leftmost image describes the goal shape; if the robot grasps the object as in the second image, it can successfully deform the object to a shape very close to the goal shape (third image). However, with the manipulation point shown in the fourth image, the shape servo task now becomes impossible (rightmost image).

We formulate the manipulation point as a function of the visual representations of the object: $x_m = f(\phi_0, \phi_g)$, where ϕ_0 is the current point cloud and ϕ_g is that of the goal shape. We propose two methods for deriving this function f.

1) Using Keypoint Detection Heuristic: We use an unsupervised keypoint detection algorithm derived from the method in [2] to identify a set of K keypoints on the point clouds $Y_0 = (u_{10}, ..., u_{k0})$ and $Y_g = (u_1 g, ..., u_k g) \in \mathbb{R}^{K\times 3}$. We define $\delta_Y = (\|u_{1g} - u_{10}\|, ..., \|u_{kg} - u_{k0}\|)$ which measures the displacement of each keypoint from the initial to the goal point cloud. We estimate the manipulation point location as the weighted average of the top M keypoints that displace the most, with weights equal to the displacements of the keypoints.

2) Using PointConv: We learn the function f as a regression problem using the same architecture as DeformerNet (Fig. 1). We modify the model to output the Cartesian position of the manipulation point (x, y, z) instead of displacement.

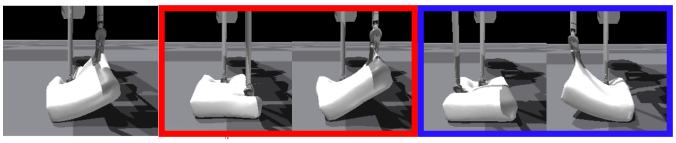


Fig. 2: Importance of manipulation point (MP) selection. Leftmost: goal shape; Red: successful MP; Purple: failed MP.

III. Experiments and Results

In the Isaac Gym environment [3], we use the bimanual daVinci surgical robot to manipulate a box object. One arm grasps one end of the object and holds it in place, while the other deforms the object into a desired shape. We created a training dataset by randomly sampling 150 pairs of object initial configurations and manipulation points. For each pair, the robot deforms the object to 200 random shapes.

Figure 3 shows the experimental results of our manipulation point selection algorithm using the two methods. We ran our Keypoint method and PointConv method with 100 test cases,

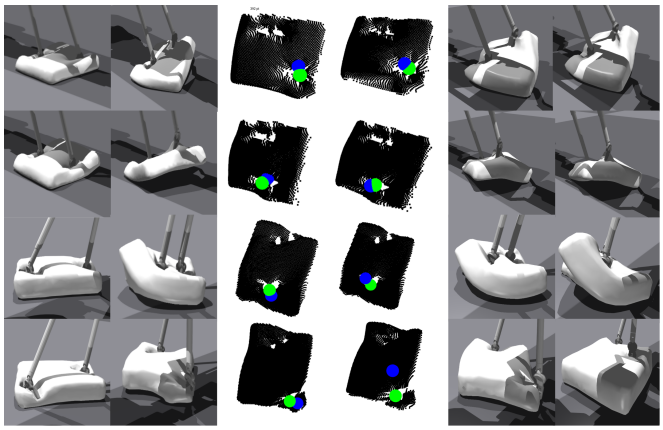


Fig. 3: Left two columns: initial and goal shapes. Center columns: manipulation points predicted by keypoint method (left) and by PointConv (right); Blue spheres are the predicted MPs and green spheres are the ground truth. Right columns: shape servo results using MPs predicted in the center column.

and the average Euclidean distance between the predicted manipulation point and ground truth was 0.035[m] and 0.036[m], respectively.

We evaluate the performance on 4 different pairs of initial-goal object shapes. Figure 3 visualizes the shape servo results. We use Chamfer distance as the evaluation metric to describe how close the final object point cloud is to the goal point cloud. Table I shows the final Chamfer distance in each scenario, using the manipulation points predicted by the two methods.

TABLE I: Shape servo results in Chamfer distance [m]

	Keypoint method	PointConv method
Case 1	0.23	0.18
Case 2	0.32	0.39
Case 3	0.26	0.53
Case 4	0.22	0.52

We compare our DeformerNet with [1], the current state-of-the-art work for learning-based 3D shape servoing. We show that the hard-coded feature vector limits the deformation function model to only learn a small set of shapes. In contrast, using learned feature vectors, DeformerNet can fit a large number of shapes and hence outperforms the method from [1].

Hu et al. [1]'s model underfits the data and results in very high train and test MSE losses (41.2 and 125.1 mm^2). Thus, the controller performs poorly and leads to very high final Chamfer distances ($>$ 1m) in all 4 test cases even with ground-truth manipulation point. In contrast, the losses of our DeformerNet are almost equal to zero (1.5 and 2.4 mm^2). Furthermore, when we train Hu *et al.*'s model on a dataset with only a few shapes, the resulted MSE losses and final Chamfer distances become much smaller. This proves that while the previous method works well when trained on only a small set of shapes, it fails to generalize on our full dataset.

REFERENCES

[1] Zhe Hu, Tao Han, Peigen Sun, Jia Pan, and Dinesh Manocha. 3-D Deformable Object Manipulation Using Deep Neural Networks. *IEEE Robotics and Automation Letters*, 4(4):4255–4261, 2019. URL https://ieeexplore.ieee.org/document/8769898.

[2] Tejas D Kulkarni, Ankush Gupta, Catalin Ionescu, Sebastian Borgeaud, Malcolm Reynolds, Andrew Zisserman, and Volodymyr Mnih. Unsupervised learning of object keypoints for perception and control. *Advances in Neural Information Processing Systems*, 32:10724–10734, 2019. URL https://arxiv.org/abs/1906.11883.

[3] Jacky Liang, Viktor Makoviychuk, Ankur Handa, Nuttapong Chentanez, Miles Macklin, and Dieter Fox. GPU-Accelerated Robotic Simulation for Distributed Reinforcement Learning. *arXiv:1810.05762*, 2018. URL https://arxiv.org/abs/1810.05762.

[4] Xiao Ma, David Hsu, and Wee Sun Lee. Learning Latent Graph Dynamics for Deformable Object Manipulation. *arxiv*, 2021. URL https://arxiv.org/abs/2104.12149.

[5] David Navarro-Alarcon and Yun-Hui Liu. Fourier-Based Shape Servoing: A New Feedback Method to Actively Deform Soft Objects into Desired 2-D Image Contours. *IEEE Trans. on Robotics*, 34(1):272–1279, 2018. URL https://ieeexplore.ieee.org/document/8106734.

[6] David Navarro-Alarcon, Yunhui Liu, J. G. Romero, and Peng Li. Model-free visually servoed deformation control of elastic objects by robot manipulators. *IEEE Trans. on Robotics*, 29(6):1457–1468, 2013. URL https://ieeexplore.ieee.org/document/6581888.

[7] David Navarro-Alarcon, Yunhui Liu, J. G. Romero, and Peng Li. On the visual deformation servoing of compliant objects: Uncalibrated control methods and experiments. *Intl. Journal of Robotics Research*, 33(11):1462–1480, 2014. URL https://journals.sagepub.com/doi/abs/10.1177/0278364914529355.

[8] David Navarro-Alarcon, Hiu Man Yip, Zerui Wang, Yun-Hui Liu, Fangxun Zhong, Tianxue Zhang, and Peng Li. Automatic 3-D Manipulation of Soft Objects by Robotic Arms With an Adaptive Deformation Model. *IEEE Trans. on Robotics*, 32(2):429–441, 2016. URL https://ieeexplore.ieee.org/document/7429768.

[9] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation. *IEEE Conf. on Computer Vision and Pattern Recognition*, 2017. URL https://arxiv.org/abs/1612.00593.

[10] Jiaming Qi, Guangfu Ma, Jihong Zhu, Peng Zhou, Yueyong Lyu, Haibo Zhang, and David Navarro-Alarcon. Contour Moments Based Manipulation of Composite Rigid-Deformable Objects with Finite Time Model Estimation and Shape/Position Control. *arXiv:2106.02424*, 2021. URL https://arxiv.org/abs/2106.02424.

[11] Radu Bogdan Rusu, Gary Bradski, Romain Thibaux, and John Hsu. Fast 3D recognition and pose using the Viewpoint Feature Histogram. *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems*, pages 2155–2162, 2010. URL https://ieeexplore.ieee.org/document/5651280.

[12] Jose Sanchez, Juan Antonio Corrales Ramon, Belhassen-Chedli Bouzgarrou, and Youcef Mezouar. Robotic Manipulation and Sensing of Deformable Objects in Domestic and Industrial Applications: A Survey. *Intl. Journal of Robotics Research*, 37(7):688–716, 2018. URL https://journals.sagepub.com/doi/abs/10.1177/0278364918779698.

[13] Wenxuan Wu, Zhongang Qi, and Li Fuxin. PointConv: Deep Convolutional Networks on 3D Point Clouds. *IEEE Conf. on Computer Vision and Pattern Recognition*, pages 9613–9622, 2019. URL https://ieeexplore.ieee.org/document/8954200.

[14] Yilin Wu, Wilson Yan, Thanard Kurutach, Lerrel Pinto, and Pieter Abbeel. Learning to manipulate deformable objects without demonstrations. *Robotics: Science and Systems*, 2020. URL https://arxiv.org/abs/1910.13439.

[15] Wilson Yan, Ashwin Vangipuram, Pieter Abbeel, , and Lerrel Pinto. Learning predictive representations for deformable objects using contrastive estimation. *arXiv preprint arXiv:2003.05436*, 2020. URL https://arxiv.org/abs/2003.05436.

APPENDIX

A. Additional Experiment Visualizations

Due to space limits, we only present the final shape servo results in the main text. Here, for each shape servo test case mentioned in the experiment, we provide key frames of the whole manipulation sequence (Fig. 5). In addition, Fig. 6 shows the Chamfer distance between the current object point cloud and the goal object point cloud over time when running our DeformerNet controller.

Figure 4 shows the experimental setup. We use the bimanual daVinci surgical robot to manipulate a soft box object which mimics human tissue. The left arm grasps one end of the object and holds it in place, while the right arm deforms the object into a desired shape.

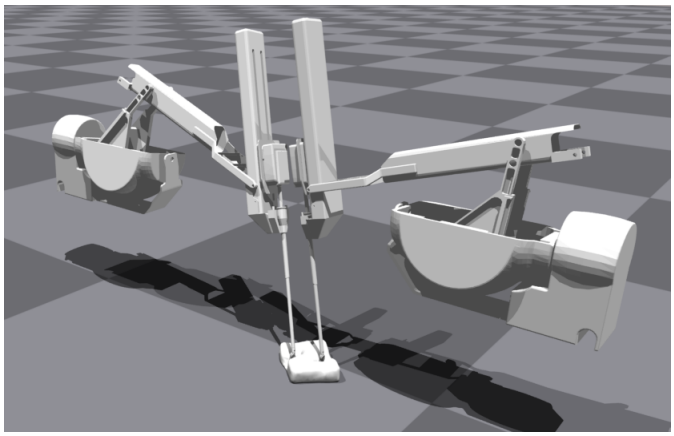


Fig. 4: Experimental setup of the two armed daVinci surgical robot in the Issac gym simulator [3].

B. DeformerNet Details

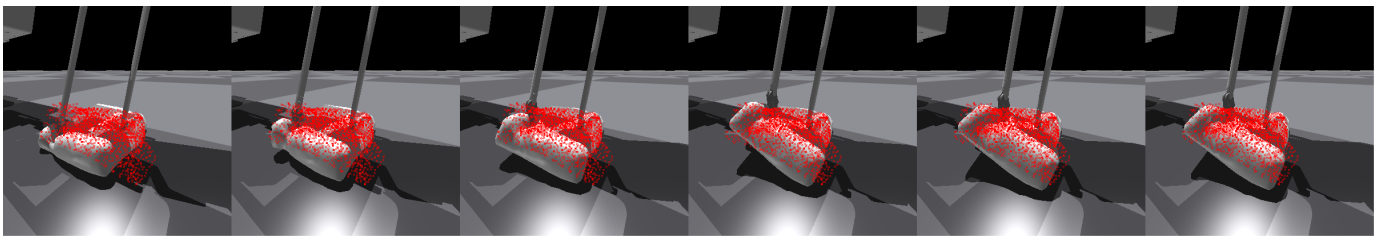
As shown in Fig. 1, DeformerNet consists of two stages: feature vector extraction and defomation control inference. In the first stage, we perform convolution on 3D point clouds to extract feature vectors representing the states of the object shapes. This stage takes two point clouds as the inputs: one of the current object shape and one of the goal object shape. The output of this stage is two 256-dimension feature vectors.

We then subtract the two feature vectors one from another and feed this to the second stage. The deformation control inference stage takes this 256-dimension *differential feature vector* and passes it through a series of fully-connected layers (128, 64, and 32 neural units, respectively). The fully-connected output layer produces the 3D manipulation point displacement. Note that this is also equivalent to the robot's end-effector Cartesian position displacement since the robot directly controls the position of the manipulation point. We use an ReLU activation function and batch normalization for all convolutional and fully-connected layers except for the output layer.

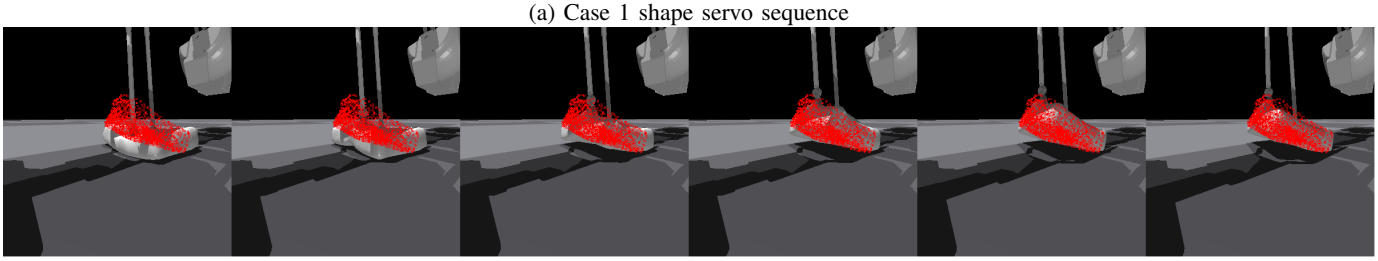
We use the standard mean squared error loss function for training our DNN. We adopt the Adam optimizer and a decaying learning rate which starts at 10^{-3} and decreases by 1/10 every 50 epochs.

C. Experiment Details

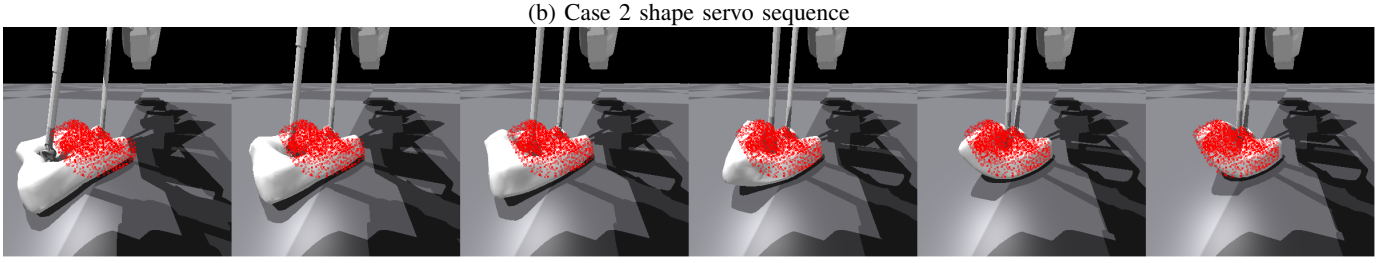
Partial point clouds are generated and segmented out from the robot and background using the depth camera available inside the Issac gym environment. We sample 2048 points on each object point cloud using the Furthest Point Sampling method from [9]. For the Keypoint Detection Heuristic method, we use 200 keypoints on each point cloud. The physical property of the object used in the experiment is: Young modulus = 1000 Pa, Poisson = 0.3.



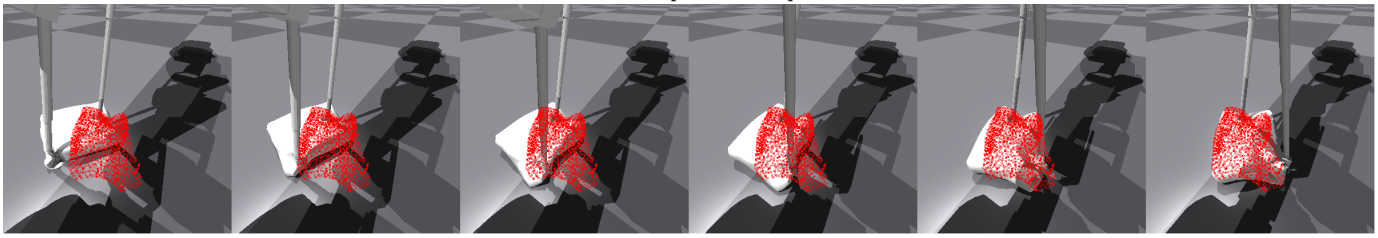
(a) Case 1 shape servo sequence



(b) Case 2 shape servo sequence



(c) Case 3 shape servo sequence



(d) Case 4 shape servo sequence

Fig. 5: Additional visualizations of the robot performing shape servoing to a variety of target shapes. The sparse red clouds visualize the target shapes of the object.

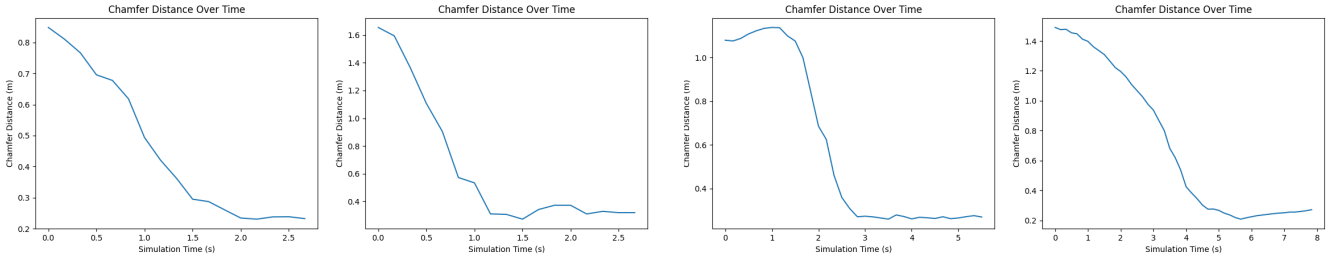


Fig. 6: Chamfer distance between the current object point cloud and the goal object point cloud over time. From left to right: cases 1, 2, 3, & 4 respectively.