Proceedings of 2022 IEEE
International Conference on Mechatronics and Automation
August 7 - 10, Guilin China

Generating Shape Transitions of Deformable Linear Objects Using Generative Adversarial Networks

Kimitoshi Yamazaki, Ryo Matsuura and Solvi Arnold
Faculty of Engineering
Shinshu University
Wakasato 4-17-1, Nagano, Nagano, Japan
{*kyamazaki, s_arnold*}@shinshu-u.ac.jp

***Abstract* - This paper presents a framework for generating shape transitions for deformable linear objects. We adopt a GAN-based approach to generate object deformations without reliance on physics simulation. We represent the deformable linear object as a point chain, and let individual shape configurations be represented as points in a low-dimensional latent space. We propose a training method for obtaining these latent representations. Then we propose a method for generating and visualizing smooth deformations. Furthermore, we propose a method for simulating how the object will deform when one of its terminal points would be moved in a given direction. We evaluate the method on the task of generating shape transitions for linear displacements of the terminal points.**

Index Terms – Deformable Linear Object, Generative Adversarial Networks, Latent space, Manipulation

I. INTRODUCTION

Tasks involving manipulation of deformable linear objects (DLOs) such as cords and cables arise in various settings, in both everyday life and manufacturing. Examples from everyday life are binding cardboard boxes with rope, and inserting the connector of a charging cable into a phone's socket. Examples from manufacturing are the wiring of wire harnesses in car factories, and the wiring of electronic appliances. Such tasks remain challenging to automate, and are still mostly performed manually. Hence, automating DLO manipulation would have a substantial impact.

The goal of this work is to construct a framework for generating shape transitions for DLOs. Consider a scenario where a robot needs to manipulate a rope. We may want to predict the shape that the rope would assume if a given manipulation were to be applied to it, or we may want to bring the rope from its current shape into a given target shape configuration. The leading approach for solving such problems is to construct a physical model of the DLO, and use this model to predict the DLO's deformations. Various such methods have been explored, and some have been applied for actual wiring work [1-4].

Methods based on physical simulation are particularly well-founded for DLO manipulation tasks. However, because the process of searching for a suitable manipulation requires large numbers of virtual manipulation trials, processing times become a substantial burden. Furthermore, manual tuning of simulation parameters is required. In consideration of these issues, we propose a method that does not depend on physics simulation. Our approach employs a Generative Adversarial Network (GAN) [5]. We map DLO shape configurations into a low-dimensional latent space, making it possible to generate deformations in this latent space.

The main contributions of this work are as follows:

- We propose a method that maps DLO shape configurations represented as point chain models into low-dimensional latent representations, and operates on shape configurations as points in a low-dimensional space. We propose a training-based method for obtaining this latent space mapping.
- We propose a method for generating and visualizing smooth DLO deformations using the aforementioned latent space. We also propose a method for simulating how the shape configuration would change under displacement of the DLO's endpoints in a given direction.

The remainder of this paper is structured as follows. In the next section we discuss related work. In Section III, we explain our objectives, task setting, and approach. Section IV explains our training method for obtaining the latent space mapping, and Section V our method for generating shape transitions. In Section VI we present evaluation experiments in simulation, and Section VII concludes the paper.

II. RELATED WORK

Automation of DLO manipulation requires capabilities for deriving and evaluating manipulation trajectories. Consequently, one of the core underlying technologies DLO manipulation builds on is DLO modeling [6]. Numerous DLO modeling strategies have been proposed over the years. One method is to represent the DLO as a mass-spring body. Wakamatsu et al. [7] proposed a modeling approach based on an extension of differential geometry. Bergou et al. [8] proposed the discrete rod model and proved their model via quantitative buckling, stability, and coupled-mode experiments, and so on. Lv et al. [9] introduced a mass-spring model for calculating cable deformations, including torsion. Roussel et al. [10] proposed DLO manipulation planning method based on kinodynamic model.

Research on DLO manipulation is numerous. Zhu et al. [11] proposed a framework that allows the robot to use contact for shaping the cable. Yamakawa et al. [12] simplify the problem of model generation by exploiting rapid hand motions, and realize robotic cord-tying. Matsuno et al. [13] automatically acquire model parameters through observation of real DLOs in static, stable shape configurations. Wakamatsu et al. [7] validated their modeling approach in a manipulation planning setting. Pai et al. [14] proposed a Cosserat rod model for DLO modeling, and applied this approach for simulation of suture

978-1-6654-0851-6/22/$31.00 ©2022 IEEE

thread. Arnold et al. [15] proposed a method for open-goal manipulation planning using Compositional Pattern-Producing Networks (CPPNs). Saha et al. [16] targeted DLO tying and untying manipulations, performing manipulation planning using a cylindrical DLO model. Chang et al. [6] tackle the task of unplugging a cable's connector from one socket and inserting it into another, performing trajectory generation on basis of physical simulation of the cable. In recent years, methods that combine differentiable simulation and reinforcement learning to allow robots to manipulate DLOs have attracted much attention. In this trend, methods for manipulating DLOs have also begun to be proposed [17, 18].

As illustrated by these examples, research on automatic DLO manipulation generally employs some type of model for representing the DLO's shape. However, using e.g. a physical model to predict deformations requires extensive tuning of the model parameters and so on. Calculating a deformation using a physical model requires that we simulate the deformation's time series. The high computational cost of accurate physical simulation poses a challenge for such approaches. In response to these current conditions, this work focuses on computational cost in particular, and aims to propose a method that can overcome this hurdle.

III. Objectives and Approach

A. Strategy for generating shape transitions

As discussed in the previous section, the dominant approaches for DLO manipulation employ numerical simulation using physical models. In this section, we step away from this paradigm, and consider the problem of learning a feature space for representing DLO shape transitions. One method for achieving this is to use an Auto-Encoder (AE), a type of neural network. For example, we could train a Convolutional Auto-Encoder (CAE) on a set of images depicting a cord in various shape configurations, and extract compressed representations of such images from the bottleneck layer of the CAE's hourglass-shaped architecture. We can say that such representations represent individual cord shape configurations as points in a low-dimensional space. Below we will refer to this low-dimensional space as a *latent space*.

On basis of the reasoning above, we explore methods for mapping DLO shape configurations to points in a latent space. A key consideration here is the need to ensure that the various DLO shape configurations are properly positioned in the latent space. First of all, it should be possible to recover a clear DLO shape configuration from a given point in latent space. Secondly, we would like for similar shape configurations to be mapped to nearby points in latent space. This is because the goal of this work is to produce shape transitions for the purpose of robotic manipulation.

Assuming that a suitable latent space can be constructed, moving a focal point through the latent space should let us obtain the corresponding deformation. This should be substantially faster than calculating the corresponding deformations through simulation. Furthermore, the problem of generating the manipulation to produce a given shape would be reduced to finding a suitable route through the latent space.

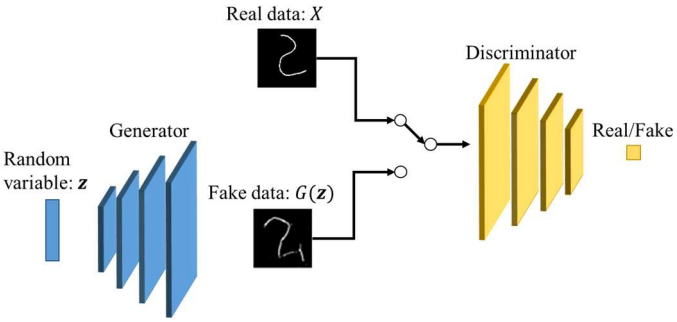


Fig. 1 Generative Adversarial Network

B. Issues and Approach

We train our latent mapping in accordance with the strategy and requirements laid out above. We introduce a Generative Adversarial Network (GAN) as illustrated in Fig. 1. Using the Discriminator network's assessment of whether a data example is real or generated, we train the Generator network to generate synthetic examples on par with data obtained from the actual DLO. If this training process is successful, we obtain a mapping from latent variable vectors z to actual DLO shapes.

Assuming the above approach, realizing the goals of this work requires that we develop functionalities for the following:

- Constructing a suitable latent space and generator
- Generating DLO shape transitions
- Finding the latent representations for given DLO shapes

The last item is necessitated by the fact that planning manipulation for an actual cord requires that we provide a start and goal shape configuration. Hence, we need to know where these shapes are located in the latent space. That is, we need a search procedure for finding the point in latent space that corresponds to a given shape configuration. Once the relevant points are found, manipulation planning can be performed by finding a curve segment that suitably connects these points. This functionality falls under the second item in the above list.

IV. Latent Space Learning

A. Generative Adversarial Networks (GAN)

Generative Adversarial Networks are a type of generative model that aims to generate novel data examples in close accordance with the data used to train the model. The model is composed of two neural networks: a *Generator network* and a *Discriminator network*. We functionally denote the Generator as $G(\cdot)$ and the discriminator as $D(\cdot)$. The Generator takes a random point from the latent space as input, and is tasked with generating a fake example $G(\boldsymbol{z})$ resembling data from real dataset X, so as to deceive the Discriminator. Conversely, the Discriminator alternatingly receives real examples $x \in X$ and fake examples $G(\boldsymbol{z})$, and is tasked with distinguishing between real and fake examples, by evaluating $D(x)$ to 1 and $D(G(\boldsymbol{z}))$ to 0. By training the networks using loss functions set in accordance with these goals, it is possible to obtain a Generator that produces clean examples indistinguishable from real data.

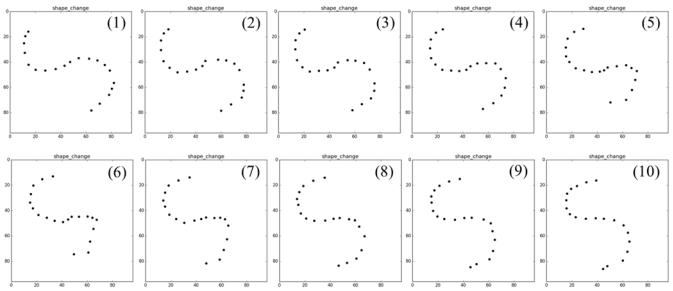


Fig. 2 Shape search in the latent space using the Encoder

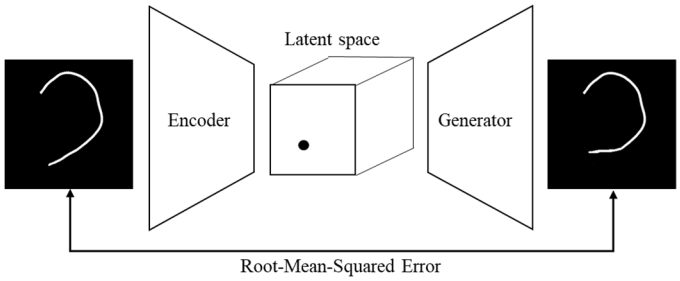


Fig. 3 Shape Search in the Latent Space Using Encoder

The Generator and Discriminator are optimized using the following Minimax loss function.

$$\min_G \max_D V(D,G) = E_{x\sim p_r(X)}[\log D(x)] + E_{\boldsymbol{z}\sim p_g(Z)}\left[\log\left(1 - D\big(G(z)\big)\right)\right] \quad (1)$$

Where $E_{x\sim p_r(X)}[\cdot]$ denotes the expected value for examples x drawn from distribution $p_r(X)$ of the set of real data, and $E_{\boldsymbol{z}\sim p_g(\boldsymbol{Z})}[\cdot]$ similarly denotes the expected value for examples $\boldsymbol{z}$ from the Generator's distribution $p_g(\boldsymbol{Z})$. Early GAN models suffered from instability in the training process, but Deep Convolutional GANs (DCGANs) [19] are comparatively stable. DCGANs incorporate a number of tweaks that have been shown to improve performance empirically, e.g. batch normalization [18], avoiding the use of fully connected hidden layers in deep architectures, and using the LeakyReLU activation function throughout the Discriminator network. This work builds on the DCGAN architecture.

B. Shape Representation using Point Chain Models

GANs have primarily been applied on images. In early stages of this work, we attempted to construct our latent space using greyscale images of a cord. However, it proved difficult to achieve proper representation of shape configurations not present in the training data. One problem was the difficulty of recovering high quality cord images. Therefore, we explored alternative representations. Eventually we settled on the more direct shape representation provided by a point chain model.

A point chain model represents a DLO as a connected sequence of positional coordinates. A cord discretized into n points is represented as $x = [\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots, \boldsymbol{x}_n]$, with each $\boldsymbol{x}_i$ denoting the coordinates of one point in the chain. Since this representation is more compact that an image-based representation, computational cost is reduced, leading to faster processing times. Furthermore, in contrast to image-based representations, 3-dimensional cords can also be represented easily. The Generator outputs sequences of point coordinates as arrays of size $n \times d$, where d denotes the coordinate dimensionality.

C. Constructing the Latent Space

As explained in Section IV-A, our approach employs a DCGAN. To improve performance on our use case, we introduce a number of modifications. First, in order to further expand the variability of the generated examples, we replace Batch Normalization in the Discriminator with Spectral Normalization [20, 21]. Spectral Normalization imposes an upper limit on convolution kernel weights by normalizing the kernel's spectral norm to 1. Its implementation is simple and its computational cost is small.

Secondly, we modify the loss function as follows. In initial experiments, we observed that the points in the point chains $G(\boldsymbol{z})$ generated by the Generator were not evenly spaced. To promote proper point spacing, we add the following loss function to the function shown in Equation (1).

$$\min_G L_d(G) = \sum_{i=1}^{n-1}\big(dr - df_i(G)\big)^2 \quad (2)$$

Here dr is the average distance between adjacent points in the training data, and df_i is the distance between two points, calculated as follows:

$$df_i(G) = \sqrt{\big(G(\boldsymbol{z})_{i,0} - G(\boldsymbol{z})_{i+1,1}\big)^2 + \big(G(\boldsymbol{z})_{i,1} - G(\boldsymbol{z})_{i+1,1}\big)^2}$$

The full loss function then becomes

$$L = \alpha \min_G L_d(G) + (1-\alpha)\min_G V(D,G) + \max_D V(D,G),$$

where α is a weight coefficient.

Using the latent space obtained with the above method, it is easy to generate continuous transitions from one shape configuration to another. We locate the latent representations of the initial shape and goal shape in the latent space, and connect them with a straight line. Then, by sampling closely spaced points on this line and mapping them to point chain representations, we obtain a continuous shape transition. Fig. 2 shows an example. First, we find the latent representations for the shapes in panels (1) and (10). Then we connect these points in latent space with a straight line, and find a set of points evenly dividing this line. Panels (2) through (9) show the point chain representations reconstructed from these points.

V. Shape Transition Generation

A. Shape Search in Latent Space

To leverage the latent space for DLO manipulation planning, we need to be able to locate a given real-world shape configuration in the latent space. We do this as follows. We assume to have a trained Generator. We couple this generator to an Encoder network, as illustrated in Fig. 3. We functionally denote the Encoder as $E(\cdot)$, and the Generator as $G(\cdot)$ as before. Input for the Encoder is the point chain representation x to be encoded, and output is the corresponding latent representation $E(x)$. The Generator receives Encoder output $E(x)$ as input

and outputs the point chain representation $G(E(x))$. While keeping the Generator fixed, we train the Encoder so as to make the Generator's output match the Encoder's input, using the following squared error loss function.

$$L(E,G) = \{x - G(E(x))\}^2 \quad (3)$$

The trained Encoder provides latent representation $\boldsymbol{z}$ of point chain representation x in the latent space. To verify that this latent representation is accurate, we feed $\boldsymbol{z}$ into the Generator to obtain x'. We calculate the mean squared error between x and x', and if this error is below a given threshold, we consider $\boldsymbol{z}$ an accurate latent representation of x.

B. Constrained Shape Transition Search

In Section IV-C, we showed how a continuous DLO shape transformation can be obtained. However, for the purpose of manipulation we need to solve the reverse problem. For example, we may want to know how the DLO's shape will change when a specific part of it is moved along a given trajectory. In this section, we explain how to generate shape transitions, using the example of linear motions applied to the DLO's endpoints. We assume to have a fully trained Generator and Encoder.

We start by defining the current and goal shape configurations. Then, using the method explained in the previous section, we find the latent representations for these shapes, and verify their accuracy. Then we generate the shape transformation for a linear motion of the DLO's endpoint using the following procedure.

1. Initialize a list containing just the latent representation of the initial shape.
2. Select a random point P from the latent space. From the list, find the point Q nearest to P.
3. Draw a line between P and Q, and find point R on the line at distance l from point P. Point R is a *candidate* for addition to the list.
4. Feed point Q and candidate point R into the Generator to obtain a sequence of point chain representations.
5. Check whether the similarity between these point chain representations exceeds a given threshold, and whether the endpoint under manipulation is moving in a straight line. If both conditions are met, candidate point R is added to the list, with Q recorded as its *parent* point.
6. If either condition is unmet, return to step 2 and continue the process with a different random point P.

The search process ends when the distance between the latent representation of the goal shape and the nearest point in the list drops below a given threshold. By tracing the sequence of parent points from this point, we obtain a sequence of latent representations connecting the initial shape to the goal shape. By feeding this sequence into the Generator, we then generate the shape transformation that occurs when a linear motion is applied to the DLO's endpoint.

VI. Verification by Simulation

A. Settings for Neural Network Training

We define the DLO as a point chain model consisting of 20 points, i.e. $n = 20$. The network architecture of the Generator is given in Fig. 4. Based on preliminary experimentation, we set the dimensionality of the latent space to 5. Latent representations are sampled from a 5D uniform distribution on the interval [-1,1]. We use the hyperbolic tangent activation function on the output layer of the Generator, and LeakyReLU on all other layers. The architecture of the Discriminator is given in Fig. 5. All layers of the Discriminator apply spectral normalization and LeakyReLU activation. For each connection weight update of the Generator, we run five updates of the Discriminator. We use a batch size of 6000 and the Adam [22] update rule with learning rate 0.0002, and first order momentum set to 0.5, training the net for 200000 epochs. Weight coefficient α in the loss function is set to 0.5.

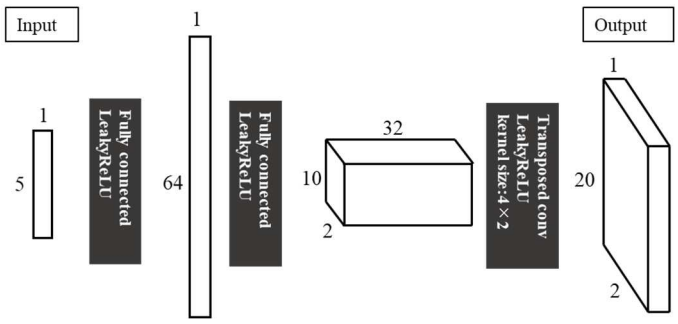


Fig. 4 Generator Architecture

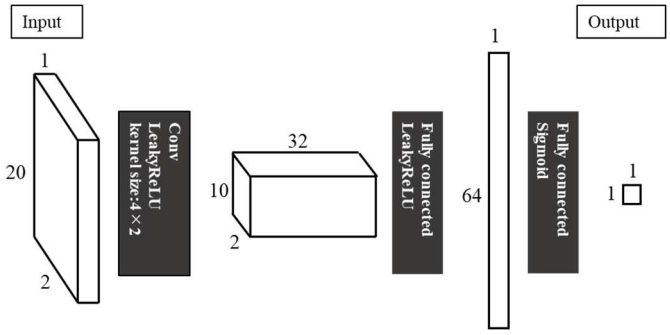


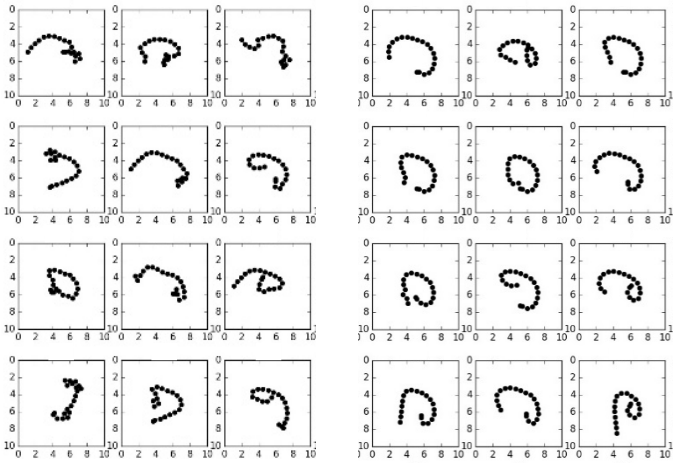
Fig. 5 Discriminator architecture

The Encoder's architecture follows that of the Discriminator, except for the size of the output layer, which is changed to 5×1 in order to output points in latent space. Training settings follow those of the Discriminator, except that the number of training epochs is reduced to 50000. The Encoder uses hyperbolic tangent activation on its output layer, and LeakyReLU on all other layers.

B. Data Collection

Data for training the GAN and Encoder was collected by means of simulation. For considerations of speed and interactivity, we used Blender [23]. We place a DLO in the simulation environment, and record its shape in point chain format while we apply linear displacements to its endpoints.

The DLO was simulated as a softbody. This formalism allows us to treat the DLO as a point chain model directly. Physical parameters were set as follows: *friction*: 10, *mass*: 2, *pull*: 0.95, *push*: 0.8, *damp*: 0.5. Other parameters (*plastic*, *bending*, *length*) were set to 0. To manipulate the DLO, we used



(a) (b)

Fig. 6 Examples of point chains created in Blender. (a) Point chains generated without cosine similarity-based filtering. (b) Point chains generated with cosine similarity-based filtering.

Blender's *empty* object. Attaching the empty object to a point on the DLO allows us to deform the DLO by moving the empty object around. However, motion in random directions can produce shapes that fail to resemble real cord. Hence we assess whether a given shape is *cord-like*, and add only shapes passing this assessment to the dataset.

The assessment is performed using cosine similarity. The main goal is to filter out shapes with overly acute curvatures. Starting from the cord's endpoint, we find the difference vector from each point to the next, and calculate the cosine similarities between subsequent vectors in this sequence. If any fall below a given threshold, the sample is rejected. Fig. 6 shows examples of point chain representations obtained with (b) and without (a) this criterion.

C. Additional Training

In Section V-B we explained latent representation search. However, if the latent space is not structured properly, this method may fail to find an accurate latent representation for a given shape. In this case, we can improve the latent space through additional training as follows.

As previously explained, the point chain model has its roots in physical simulation. Hence, it is easy to instantiate a point chain representation as a physical cord in simulation. From a cord shape we wish to incorporate into the latent space, we produce a set of slightly deformed variations, and perform additional training of the GAN and Encoder with this data. Then, we again search the latent space for the corresponding latent representation.

This improvement process is performed offline, but over time it makes it possible to construct a practically applicable latent space.

D. Shape Transition Search

Figure 7 shows an example of point chain representations recovered from given latent representations. For shapes from the training data, and shapes closely resembling these, we find that uninterrupted point chain representations are recovered from

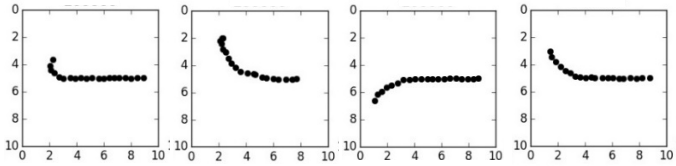


Fig. 7 Examples of generated point chains.

the corresponding latent representations. This shows that it is possible to construct a latent space in which point chain representations of the cord are distributed as latent representations. Furthermore, it shows that it is possible to construct a Generator that can reconstruct uncompressed, high-dimensional shape information from compressed representations of point chains.

Next, we experimentally evaluated our method for generating shape transformations. With the cord in a given shape configuration, we move one endpoint by some distance in a straight line, and let the pre- and post-displacement shapes be the initial and goal shape for transformation generation. The threshold for cosine similarity introduced in Section VI-A is set to 0.25. We perform additional training as described in Section V-C for 100 epochs. The threshold for the mean squared error of Equation (2) is set to 0.2.

Figure 8 shows three examples of generated shape transformations. We observe that appropriate intermediate shapes are generated in each example. These results indicate that functionality for finding the latent representation of a given point chain was constructed successfully for part of the latent space. However, we also observed cases where the latent representation of a given point chain could not be found, even when additional data collection and training was performed. We hypothesize that such failures could be avoided by training the GAN and Encoder on a wider variety of shape configurations in the initial training phase. Another factor may be the number of training iterations. In this experiment, we performed 100 epochs of additional training of the GAN and Encoder. We confirmed proper convergence for the Encoder by graphing the loss curve. However, the loss function for the GAN provides no clear indication of how well the Generator's output approximates the training data, so whether the number of training iterations is sufficient for the GAN remained unclear. Addressing this issue will require a method for better assessing a GAN's training progress.

VII. Conclusions

We introduced a framework for generating shape transformations for Deformable Linear Objects, using a GAN to generate deformations without reliance on physical simulation. We represent DLO shape configurations as points in a low-dimensional latent space, and generate shape transformations within that space. We confirmed that this method can generate arbitrary DLO shape by sampling a point in the latent space and then decoding it. We presented a method to find the latent expression of the specified shape from the latent space. Finally, we confirmed shape transformations for linear displacements applied to a DLO's endpoints.

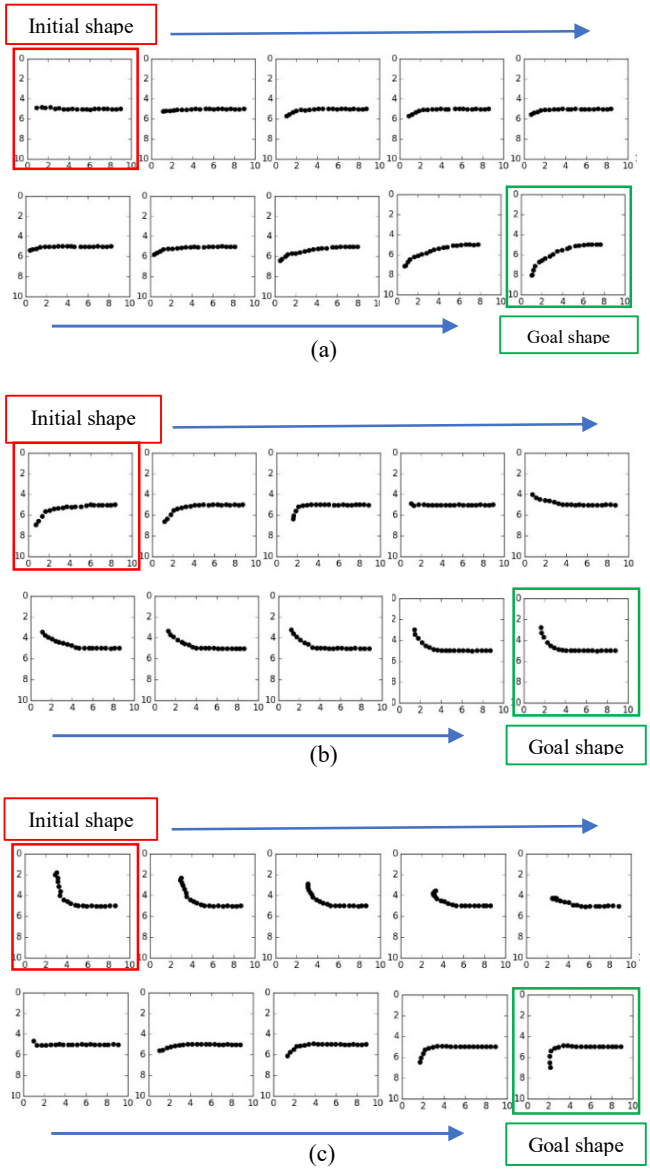


Fig. 8 Examples of shape transition search.

In future work, we aim to improve our training methods in order to construct a more effective latent space, and expand the variety of shape transformations that can be generated. We also hope to introduce functionalities for handling DLO rigidity and kinks. Then, we will realize a real-world DLO manipulation task through experiments using actual robots, and make wide use of the proposed approach.

ACKNOWLEDGMENT

This work was partially supported by JST KAKENHI and JST [Moonshot R&D][Grant Number JPMJMS2034]. .

REFERENCES

[1] X. Jiang, K.-M. Koo, K. Kikuchi, A. Konno, and M. Uchiyama, "Robotized assembly of a wire harness in car production line," IEEE/RSJ Int. Conf. on Intelligent Robots and Systems, pp. 490-495, doi: 10.1109/IROS.2010.5653133, 2010.

[2] F. Yumbla et al., "Preliminary Connector Recognition System Based on Image Processing for Wire Harness Assembly Tasks," 20th Int. Conf. on Control, Automation and Systems, pp. 1146-1150, 2020. doi: 10.23919/ICCAS50221.2020.9268291

[3] C. Ying, Y. Mo, Y. Matsuura, K. Yamazaki, "Pose Estimation of a Small Connector Attached to the Tip of a Cable Sticking Out of a Circuit Board," International Journal of Automation Technology, Vol. 16, No. 2, pp. 208-217, 2022. https://doi.org/10.20965/ijat.2022.p0208

[4] K. Sano, S. Iijima, and K. Yamazaki, "A Case Study on Automated Manipulation for Hooking Wiring of Flexible Flat Cables," 2019 IEEE Int. Conf. on Mechatronics and Automation, pp. 793-798, 2019. doi: 10.1109/ICMA.2019.8816286

[5] I. Goodfellow, J. Pouget-Abadie. M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, Y. Bengio, "Generative Adversarial Nets,". Proceedings of the International Conference on Neural Information Processing Systems, pp. 2672–2680, 2014.

[6] P. Chang, T/ Padir., "Model-Based Manipulation of Linear Flexible Objects: Task Automation in Simulation and Real World," Machines 2020, vol. 8, no. 3, 46, 2020.

[7] H. Wakamatsu and S. Hirai, "Static modeling of linear object deformation based on differential geometry," Int. J. Robot. Res., vol. 23, no. 3, pp. 293–311, 2004.

[8] Bergou, M, Wardetzky, M, Robinson, S, Audoly, B, Grinspun, E, "Discrete elastic rods," ACM Transactions on Graphics 27(3): 1–12, 2008.

[9] N. Lv et al., "Physically based real-time interactive assembly simulation of cable harness," Journal of Manufacturing Systems, vol. 43, no. 3, pp. 385-399, 2017

[10] O. Roussel and M. Taïx, "Deformable linear object manipulation planning with contacts," in Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems, 2014.

[11] J. Zhu, B. Navarro, R. Passama, P. Fraisse, A. Crosnier and A. Cherubini, "Robotic Manipulation Planning for Shaping Deformable Linear Objects WithEnvironmental Contacts," in IEEE Robotics and Automation Letters, vol. 5, no. 1, pp. 16-23, Jan. 2020, doi: 10.1109/LRA.2019.2944304.

[12] Y. Yamakawa, A. Namiki, and M. Ishikawa, "Dynamic high-speed knotting of a rope by a manipulator," International Journal of Advanced Robotic Systems, Vol. 10, No. 10, p. 361, 2013.

[13] T. Matsuno and T. Fukuda, "Manipulation of flexible rope using topological model based on sensor information," In 2006 IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 2638–2643, 2006.

[14] D. K. Pai, "Strands: Interactive simulation of thin solids using cosserat models," In Computer Graphics Forum, Vol. 21, pp. 347–352. Wiley Online Library, 2002.

[15] S. Arnold and K. Yamazaki, "Implicit Policies for Deformable Object Manipulation with Arbitrary Start and End States: A Novel Evolutionary Approach," in Proc. of IEEE Int'l Conf. on Robotics and Biomimetics, pp. 1776 – 1781, 2016.

[16] M. Saha and P. Isto, "Manipulation planning for deformable linear objects," IEEE Transactions on Robotics, Vol. 23, No. 6, pp. 1141–1150, 2007.

[17] X. Ma, D. Hsu, W. S. Lee , "Learning Latent Graph Dynamics for Deformable Object Manipulation," ICRA Workshop on Representing and Manipulating Deformable Objects, 2021

[18] Fei Liu, Entong Su, Jingpei Lu, Mingen Li, Michael C. Yip, "Differentiable Robotic Manipulation of Deformable Rope-like Objects Using Compliant Position-based Dynamics," CoRR, 2022.

[19] A. Radford, L. Metz, and S. Chintala, "Unsupervised representation learning with deep convolutional generative adversarial networks," arXiv preprint arXiv:1511.06434, 2015.

[20] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," CoRR, Vol. abs/1502.03167, 2015.

[21] T. Miyato, T. Kataoka, M. Koyama, and Y. Yoshida, "Spectral normalization for generative adversarial networks," arXiv preprint arXiv:1802.05957, 2018.

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

[23] blender. https://www.blender.org/