Non-Prehensile Manipulation Learning through Self-Supervision

Ziyan Gao, Armagan Elibol, and Nak Young Chong
School of Information Science
Japan Advanced Institute of Science and Technology
Nomi, Ishikawa, Japan
{s1920013, aelibol, nakyoung}@jaist.ac.jp

***Abstract*—Manipulation is one of most emerging research and development areas in the field of robotics. Recently, state representation learning for control has been gaining attention. In this paper, we proposed a novel learning model based on neural networks in order to sample the actions of the robot to push objects to desired positions. Furthermore, an intuitive method was proposed to enable the robot to collect training data in an efficiently way. Specifically, a fully convolutional network encodes observations into latent space, and a mixture density network is implemented to infer an action distribution, since there are an infinite number of possible actions that may result in the same change of the state of the object. Through extensive experimental simulations and comparisons with the existing models, we demonstrated the efficiency of the proposed method applied to non-prehensile manipulation, such as pushing or rotating of small objects on the table.**

Index Terms– **non-prehensile manipulation, state representation learning, fully convolutional autoencoder, mixture density network**

I. Introduction

Object manipulation skills are important for humans in their daily life as well as many other fields such as manufacturing and service industries. Robotic manipulation has an important role in many of these fields since, robots can help humans to direct their efforts to more skill-needed works instead of tedious and repetitive works. Also, robots improve efficiency without having any distractions. Human beings can manipulate objects with a small error, even they do not know the object characteristics. In [1], authors inferred that it might be due to the humans' internal model of physics which enables them to understand the physical properties of objects and to predict their dynamics under the action of external forces. It is also discussed in [16] that spending years playing with objects helps infants to develop their own internal models. For a robot, however, it is pretty difficult to manipulate a novel object. In order to perform manipulation tasks successfully, a robot must be able to detect the related features, such as the position and orientation (pose), of the object under the current scene. Moreover, the robot needs to understand the changes in the pose of the object, which is even more challenging.

The fully convolutional neural network was proposed in [10] and it has been successfully applied to the semantic segmentation problem. It offers several benefits, such as no limitation on the input size and preserving the spatial information. In this work, we apply a fully convolutional autoencoder (FCA) to transform the depth image into a low-dimensional latent space. The FCA encoder uses the same weights to encode each patch of input image individually. The size of the patch depends on the size of the feature map included in latent space. FCA also has the spatial-invariant property, which results that FCA could encode the object properly regardless of its position in the scene. Similar to the problem of robot grasp synthesis, in the case of non-prehensile manipulation of objects, pushing objects faces the same problem that there is an uncountable number of possible solutions. An example scenario is illustrated in Fig. 1. The objective is to push the object to the desired state (shown in the second image), and humans may push it in many different ways. Recently, inverse and forward models have been used more and more in the coding of sensorimotor simulations [14]. Inverse models allow the system to determine the motor commands necessary to achieve the desired state, while forward models predict the expected sensory feedback of a motor command, allowing rapid error detection when the actual and predicted feedback are ill-matched [3].

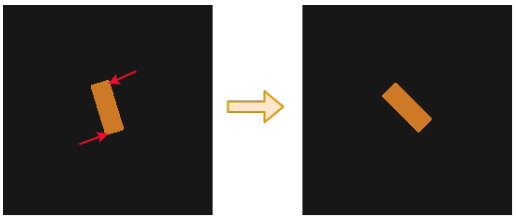

Fig. 1. The left image depicts the current position and orientation of the object while the object is in the desired state in the second image. Red arrows show some of the possible actions that impose the same effort to the object. It should be noted that there might be infinite number of possible actions.

In this paper, we present a novel non-prehensile robotic manipulation method that makes use of depth images obtained from a down-looking sensor to model the change in the pose of an object. An FCA was used to transform depth images into latent feature maps. Also, Mixture Density Network (MDN) was used to be an inverse model, and the forward model was implemented in order to regularize the inverse model. Experimental validations were carried out using V-REP simulation environment [13]. The total number of 12,000

training samples were collected in 20 hours in the simulation environment. 9 objects of different shapes (see Fig. 4) were used for collecting data, and 5 novel objects were used for evaluating the performance of our method. The rest of the paper is organized as follows: The next section is devoted to outlining some of the related works. Section. III-B presents our proposed model, and in Section. IV we present experimental evaluations and performance comparisons in different test scenarios. The last section draws conclusions and describes future research directions.

II. Related Work

State representation learning (SRL) is a particular case of feature learning in which the features to be learned are low dimensional, evolve through time, and are inuenced by actions or interactions [9]. The recent state-of-art methods on SRL for control problems have been reviewed in [9]. Many of the existing works have used neural networks as an SRL model. Auto-encoders(AEs) have been applied to reduce the dimension of state to low-dimension vector. Mattner *et al.* [12] used AE to encode the 2-dimensional state from real images and used this 2D vector to balance pole. However, our goal is to manipulate novel objects. There is a need for more features such as the shape of objects, pose, and similar others. van Hoof *et al.* [17] assumed that the encoded low dimensional space is linear, and the transition is also linear. However, in the case of dynamics of object motion caused by robot pushing, it becomes non-linear. Agrawal *et al.* [1] implemented a joint training of both the forward model and inverse model by summing over the loss of both models. Also, Duan *et al.* [4] proposed a method that can combine the observation, the forward model, and the inverse model jointly. However, they only considered that the inverse model predicts a single output, which cannot be applied to real situations especially in the case of pushing motion. Different objective functions for SRL were proposed in [8], [7], [6], [15], where their performance was shown under certain circumstances. However they are most likely to face over-fitting problems when there is unavailability of abundant data. Finn *et al.* [5] proposed another model called spatial autoencoder which automatically acquires related features in a data-efficient way. However, it still requires the robot to interact with environment for several hours. Chopra *et al.* [2] proposed a neural network (called Siamese networks) which consists of more than one network that shares parameters with each other. This model can be applied to the scenario of distinguishing different inputs. In this work, we used the structure proposed in [2] to transform the depth image into latent feature maps, and developed an inverse and forward model based on the latent feature maps. Zeng *et al.* [18] underlines that skillful manipulator benefits from versatile actions such as combining non-prehensile and prehensile actions. They proposed a dual model that predicts both push and grasp actions to improve the success rate of grasping. However, our purpose is to develop a model to push the object to the desired pose, which needs to model the state change of the object and infer proper actions.

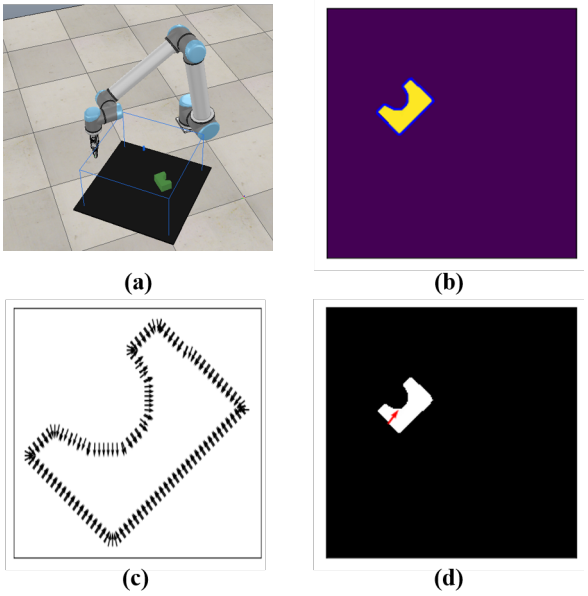

Fig. 2. (a) The simulation environment; the black plane is the working space of UR10 and a down-looking depth camera is set on the top of the black plane. (b) the captured image is processed by Eq. (2) and the contour is extracted. (c) The normal vector of each pixel on the contour is computed. (d) a normal vector is sampled from normal vectors shown in (c) and will be executed by the robot.

III. Method

In this paper, the current and target states given by depth images were used as the input of the proposed model. There is no restriction on how the objects move from the current to the target state and there are an infinite number of possible trajectories of the object motion. We assume that all the motion of the robot is linear. Given a target and an initial state, there is/are only one or multiple linear motion(s) which transforms the initial state to the target. The velocity of the robot's gripper is assumed to be slow and steady enough not to cause any sudden changes in the state. Only the motion in the XY plane is considered and Z coordinate of the working space of the robot is fixed.

A. Action Sampling

To overcome the problem (similarly reported in [1]) of having many actions that do not change the object state, we apply normalization on the depth values using Eq. 1.

$$I = |I - max(I)|, \tag{1}$$

where I is a depth image. The object contour is extracted by using the marching squares algorithm [11]. A window of size 5 is used to compute the orientation of each pixel, therefore the orientation of each pixel has 25 possible values. The orientation is computed using Eq. 2.

$$\theta = arctan2(Ct_{(i+2)} - Ct_{(i-2)}), \tag{2}$$

where Ct is a set of (x, y) coordinates of contour pixels in metric and i refers to a location of single pixel on the contour.

$$N = [cos(\theta + \pi/2), sin(\theta + \pi/2)]^T, \tag{3}$$

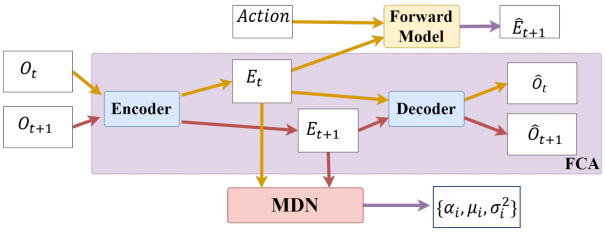

Fig. 3. The pipeline of the proposed model: The blocks with color background are learning modules, O_t and O_{t+1} are inputs, while E_t and E_{t+1} are encoded latent feature maps. *Action* is labeled action defined by Eq. 5.

where N is the normal of the curve for each pixel on the contour. As a final step, the homogeneous transformation shown by Eq. 4, in which (T) transforms the coordinate of the image into the coordinate of the robot system. Fig 2 illustrates the whole procedure.

$$[x_r, y_r, 1]^T = T_{3\times3} \cdot [x_i, y_i, 1]^T, \tag{4}$$

where x_r, y_r is the location with respect to the robot coordinate and x_i, y_i is the location in the image coordinate frame. We use homogenous transformation in 2D as we only consider the motion in XY plane.

An action is defined by Eq. 5:

$$A = \{X_s, Y_s, X_e, Y_e\}, \tag{5}$$

where X_s and Y_s represent the initial horizontal and vertical coordinates, while X_e and Y_e represent the final one, respectively.

B. Model

The pipeline of our model is given in Fig 3, and the detailed illustration of the proposed model is given in Fig 5. An FCA is used to extract the features from the depth image. Our model does not make use of any pre-trained layers from other models. The current image, the action, and the changed image caused by the gripper motion constitute one training example. Inputs are two time-consecutive images and they are fed into two identical networks. The networks encode the current observation and target observation, respectively. Since there are an infinite number of possible actions that can cause the same motion of the object, also in the phase of data collection, it is inevitable to collect different actions which cause the same motion of the object. An MDN given in Eq. 6 is used to infer a distribution of the action.

$$P(a|s_t, s_{t+1}) = \sum_{c=1}^{C} \alpha_c \mathcal{D}(\mu(s_t, s_{t+1}), \sigma(s_t, s_{t+1})), \tag{6}$$

where c denotes the index of the corresponding mixture component. There are up to C mixture components, α_c depends on the input and the sum of all α_c is one. $\mathcal{D}$ denotes the distribution to be mixed. In this work, we used Gaussian distribution determined by μ and σ.

The usage of MDN provides several benefits; Firstly, it can help to prevent overfitting due to the limited training samples, infinite number of objects, and possible actions. Secondly, learning the distribution of motion is easier for the neural network than learning a single accurate action. Thirdly, MDN can help to avoid stuck in a certain state, as it predicts diverse actions for the same circumstance. Finally, again due to the limited training data examples, existing models tend to fall into a situation called "trap". However, since MDN samples actions, the actions are different even for the same input.

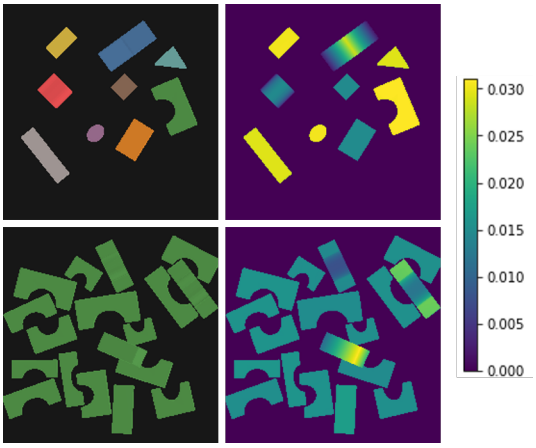

Fig. 4. The left two images are RGB images and the right ones are depth images. The size and the ratio between height, width, length of objects were changed randomly. It should be noted that depth images have been normalized.

C. Training

In the phase of training, in the first step, we only trained FCA as an initialization, then we integrated the pre-trained FCA with MDN and forward model. The following equations (Eqs. 7-10) define the loss functions.

$$L_1 = L(s_t, \hat{s_{t+1}}, W_{enc}, W_{forward}) \tag{7}$$

$$L_2 = L(a, \hat{a}, W_{enc}, W_{mdn}) \tag{8}$$

$$L_3 = L(o_t, \hat{o_{t+1}}, W_{enc}, W_{dec}) \tag{9}$$

$$L = \lambda(L_1 + L_2) + L_3 \tag{10}$$

In the above equations, L_1 is the loss of forward model which acts as a regularizer to inverse model. L_2 is detrimental log-likelihood loss which could less than zero. L_3 is the reconstruction error. W_{enc} and W_{dec} are parameters of FCA, W_{mdn} is the parameters of the MDN. We set $\lambda = 10^{-4}$ during the training phase to avoid overfitting.

IV. Experiment Results

In this work, V-REP simulator [13] is used as the simulation environment. UR10 robot is used as the agent to interact with objects. The working space is 0.512×0.512 in a horizontal and vertical direction. We used a resolution of 224×224 image to capture the object on the plane. The gripper is always in close status and the inverse kinematics is calculated by the simulator software using the pseudo-inverse method. We set the coordinates of the gripper is the same as the coordinates of the plane.

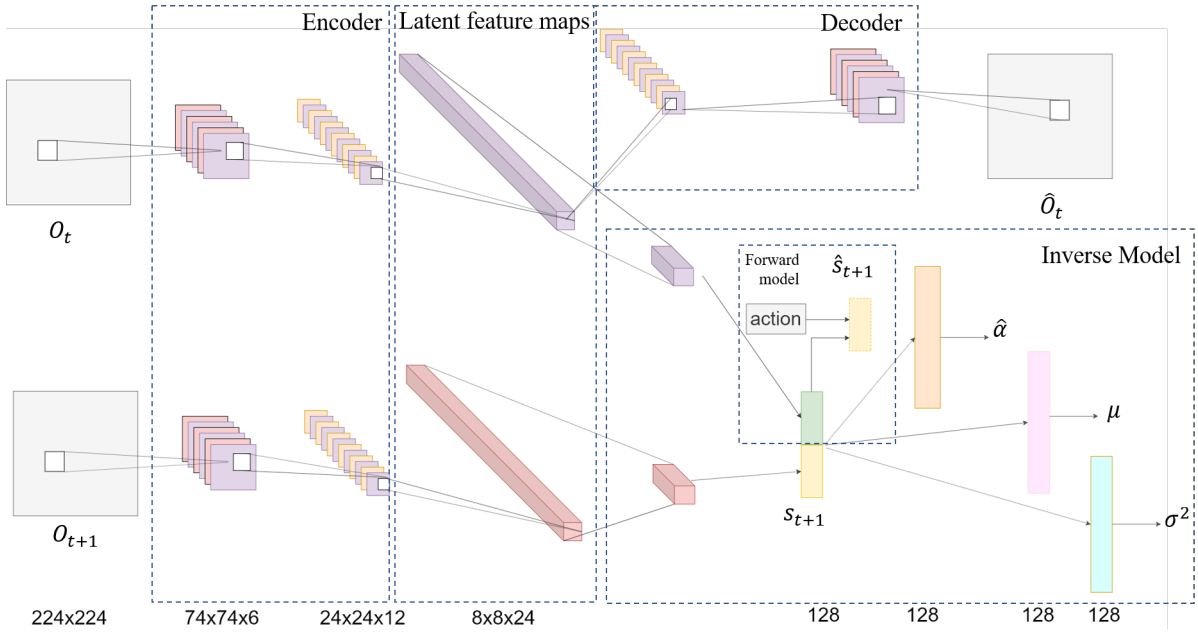

Fig. 5. O_t and O_{t+1} are consecutive time step depth images and they are processed by three convolutional layers. The size of encoded latent space is $8 \times 8 \times 24$, followed by the decoder of fully convolution autoencoder and the mixture density network. The decoder reconstructs the input based on the latent feature maps, and mixture density network predicts a mixed multivariate Gaussian distribution of the action. There is also a forward model which predicts the state of next time step based on the current state and action.

The data used in this work is acquired by a robot without human intervention. First, the object is added to the scene, and then a push action is sampled to interact with the object. Finally, the initial image, the action, and the image after exerting the action are collected to be a single training example. There were 9 objects (see Fig. 4) of different shapes and sizes considered in the experiment. In order to improve the generalization of the model, we randomly change the size and the ratio of the height, width, and length of the object. Since the depth image can describe the shape of the object more accurately, we choose the depth image to be the observation for the object. In this paper, we only consider a single object.

We used training data of size 64 as a mini-batch to train the FCA and MDN. Adam optimizer is used to minimize the loss function, and the learning rate is set to 1×10^{-3}. After 200 epochs, the optimization procedure of FCA was ended. The encoder part of FCA is used to transform the input high-dimensional data into low dimension latent feature maps. For the training of the MDN, we only ran 30 epochs as the loss function did not reduce further.

To evaluate our model, 5 different objects which were not used during the training phase were used. The objects are shown in Fig. 6. The relative position error, which is the ratio of the distance between the initial object position and target object position and the distance between object position after pushing by the robot and target object position, was used to evaluate our model. Fig. 7 illustrates the relative position error after pushing the objects; each plot in the figure demonstrates the relative position error when the robot interacted with a certain object. There are 4 different models compared, and our

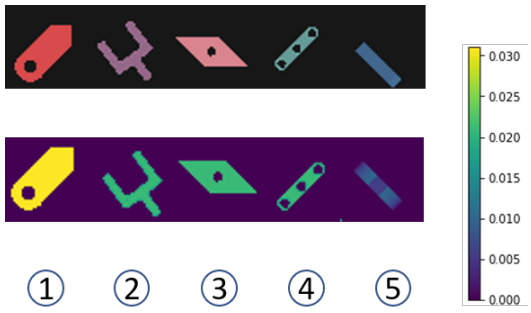

Fig. 6. The RGB image on the top illustrates 5 novel objects, while the bottom one is the corresponding depth image. It should be noted that depth images have been normalized.

proposal is the second one in Fig. 7. We repeat each experiment 100 times, and, for each time the robot was allowed to push the object at a maximum of 15 times. After each pushing step, the relative position error was computed. By comparing with other models, we found that our model performs well for each experiment: the mean of relative position error is about 0.3, while other models performed unstable. Notably, the 4th model shown in Fig. 7 performed even better than ours when pushing object1 and object2, however performed worst when pushing other objects. For the first 3 objects, the performance of our model is similar compared to others, while for the 4th and 5th objects, our model outperforms other models. This might be because the 4th and 5th objects are very small, which introduces more challenges for the robot to interact. In such cases, it is more likely to fall into sub-optima

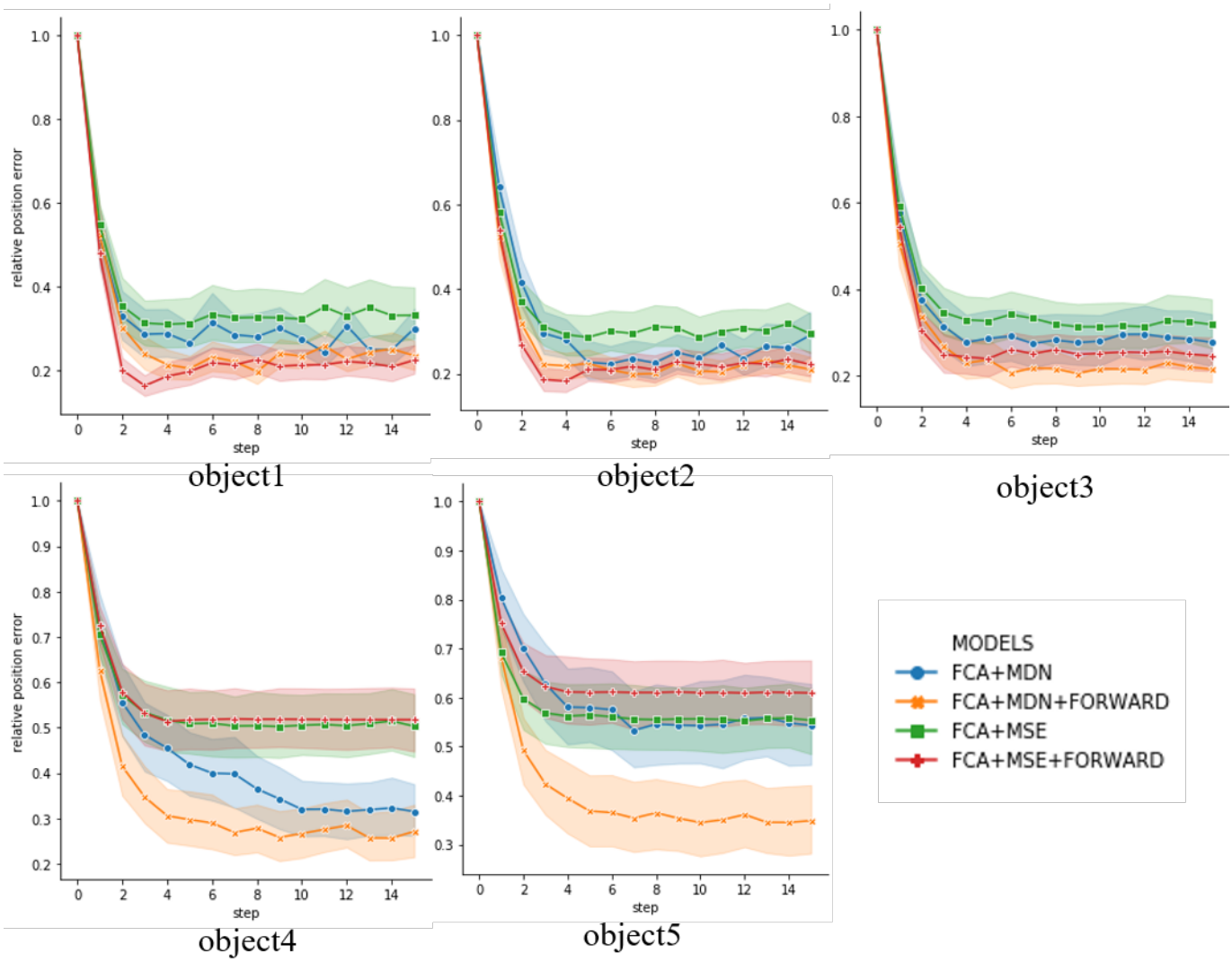

Fig. 7. Each figure shows the relative position error after every pushing action. Five figures correspond to five objects. We repeat each experiment 100 times and, for each experiment, the robot pushed object 15 times.

Models		Object1				Object2				Object3				Object4				Object5			
		mean	std	max	min	mean	std	max	min	mean	std	max	min	mean	std	max	min	mean	std	max	min
fca+mse	Step3	0.31	0.27	1.0	0.04	0.31	0.24	1.0	0.01	0.35	0.27	1.0	0.03	0.53	0.34	1.0	0.03	0.57	0.32	1.0	0.04
	Step6	0.33	0.3	1.22	0.02	0.3	0.24	1.0	0.02	0.34	0.28	1.0	0.01	0.51	0.33	1.0	0.04	0.56	0.33	1.0	0.03
	Step9	0.33	0.3	1.22	0.04	0.31	0.24	1.0	0.03	0.31	0.27	1.0	0.01	0.5	0.34	1.0	0.04	0.56	0.33	1.0	0.03
fca+mse+forward	Step3	**0.16**	**0.12**	0.58	0.01	**0.19**	**0.14**	0.81	0.01	0.25	0.23	1.0	0.04	0.53	0.32	1.0	0.02	0.62	0.33	1.0	0.06
	Step6	0.22	0.17	1.01	0.03	0.21	0.14	0.78	0.03	0.26	0.2	1.0	0.03	0.52	0.33	1.0	0.02	0.61	0.33	1.0	0.03
	Step9	0.21	0.18	1.01	0.02	0.23	0.16	0.81	0.02	0.25	0.2	1.0	0.01	0.52	0.33	1.0	0.02	0.61	0.33	1.0	0.03
fca+mdn	Step3	0.29	0.28	1.28	0.01	0.3	0.24	1.41	0.01	0.32	0.3	1.75	0.02	0.48	0.33	1.24	0.04	0.63	0.39	1.57	0.01
	Step6	0.32	0.29	1.68	0.01	0.22	0.19	1.41	0.01	0.29	0.29	1.75	0.02	0.4	0.31	1.23	0.01	0.58	0.4	1.57	0.01
	Step9	0.3	0.22	1.09	0.01	0.25	0.21	1.41	0.02	0.28	0.28	1.75	0.01	0.34	0.3	1.21	0.03	0.54	0.42	1.73	0.03
fca+mdn+forward	Step3	0.24	0.18	0.97	0.01	0.22	0.17	0.81	0.01	0.27	0.25	1.33	0.02	0.35	0.3	1.0	0.01	0.42	0.36	1.84	0.03
	Step6	0.23	0.17	0.82	0.02	0.21	0.15	0.76	0.0	0.21	0.19	1.32	0.01	0.29	0.28	1.12	0.01	0.36	0.36	1.84	0.02
	Step9	0.24	0.17	0.82	0.02	0.22	0.16	0.68	0.03	**0.21**	**0.16**	1.18	0.03	**0.26**	**0.25**	1.12	0.01	**0.35**	**0.36**	1.84	0.02

TABLE I

IN THIS TABLE, WE SHOW QUANTITATIVE RESULT FOR SOME STEPS. IT SHOWS MEAN, STANDARD DEVIATION, MAXIMUM AND MINIMUM OF RELATIVE POSITION ERROR OF 100 TIMES TRIALS.

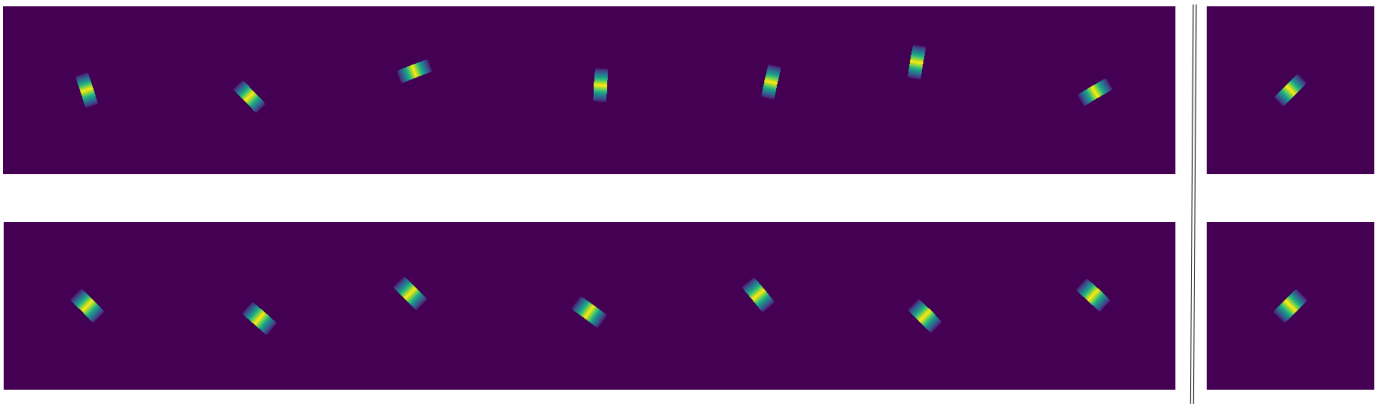

Fig. 8. There are two examples to show the performance in the case of pure rotation motion. The object in the target state is obtained by rotating the initial state 90 degrees. The upper row is the result of the model with mixture density network, and the other is the one which predicts exact actions for rotating the object. The right two images depict the target state of the object, while the first left two images depict the initial state of the object.

resulting predicted actions without any object interaction. In order to handle such situations, models need to be trained with a substantial amount of training samples. Our model, thanks to its MDN, outputs diverse actions even in the same circumstances. This helps to handle such cases efficiently. When comparing our model with the ones without the forward model (FCA+MDN and FCA+MSE), It can be seen that our model performed better than the other models. Therefore, it is noted that the forward model can help FCA to learn a better state than the ones without it. Table I shows the quantitative result of this experiment, illustrating that our model performs well for each object.

Fig. 8 shows an exemplary result of the case of a pure rotation of a known object. It can be seen that our model completed this task in several steps. However, the other model predicting only one action failed and it just sway the object without rotating.

Our model predicts a mixed Gaussian distribution of action and command the robot to push novel object to the desired position and orientation. Our method shows its advantage of predicting diverse actions to avoid falling into local optima. Besides, Fig. 8 shows that our methods performs better in the case of pure rotating the object, since, in training phase, the model tends to average the error that could cause inaccurate action prediction.

V. Conclusions and Future Work

Manipulation is one of the most important and developing abilities of robots. Moving an object to the desired position and orientation is a challenging task for a robot. Doing such tasks with a novel object that has been neither defined nor introduced to a robot previously is even more challenging. In this work, we proposed an efficient learning model to create a sequence of actions in order to push a novel object to the given target position and orientation. We extended the work of [1] and proposed a new method that integrates fully convolutional auto-encoder to dimension reduction and mixture density network to select proper actions. This model predicts a mixed Gaussian distribution for pushing objects to the desired position and orientation. We presented the efficiency of the proposed model through extensive simulations with a variety of conditions. Also, we proposed an efficient way to collect training data, which can improve the efficiency of self-exploration.

In this work, we did not consider the impact of the past actions on the current action. Furthermore, for push multiple objects in the same scene to the desired pose, it will be much more challenging, because the pose of objects can not only be affected by the pushing action of robot, but also by the motions of other objects. Our future efforts will be aimed at addressing the aforementioned issues.

References

[1] Pulkit Agrawal, Ashvin V Nair, Pieter Abbeel, Jitendra Malik, and Sergey Levine. Learning to poke by poking: Experiential learning of intuitive physics. In *Advances in Neural Information Processing Systems*, pages 5074–5082, 2016.

[2] Sumit Chopra, Raia Hadsell, Yann LeCun, et al. Learning a similarity metric discriminatively, with application to face verification. In *CVPR (1)*, pages 539–546, 2005.

[3] Richard P. Cooper. Forward and inverse models in motor control and cognitive control. In *Proceedings of the Symposium on AI-Inspired Biology*, pages 108–110. The Society for the Study of Artificial Intelligence and the Simulation of Behaviour, London, UK, 2010.

[4] Wuyang Duan. Learning state representations for robotic control. Master's thesis, Delft University of Technology, 2017.

[5] Chelsea Finn, Xin Yu Tan, Yan Duan, Trevor Darrell, Sergey Levine, and Pieter Abbeel. Deep spatial autoencoders for visuomotor learning. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 512–519. IEEE, 2016.

[6] Rico Jonschkowski, Roland Hafner, Jonathan Scholz, and Martin Riedmiller. Pves: Position-velocity encoders for unsupervised learning of structured state representations. *arXiv preprint arXiv:1705.09805*, 2017.

[7] Varun Raj Kompella, Matthew Luciw, and Juergen Schmidhuber. Incremental slow feature analysis: Adaptive and episodic learning from high-dimensional input streams. *arXiv preprint arXiv:1112.2113*, 2011.

[8] Brenden M Lake, Tomer D Ullman, Joshua B Tenenbaum, and Samuel J Gershman. Building machines that learn and think like people. *Behavioral and brain sciences*, 40, 2017.

[9] Timothée Lesort, Natalia Díaz-Rodríguez, Jean-Franois Goudou, and David Filliat. State representation learning for control: An overview. *Neural Networks*, 108:379–392, 2018.

[10] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3431–3440, 2015.

[11] William E. Lorensen and Harvey E. Cline. Marching cubes: A high resolution 3d surface construction algorithm. In *Proceedings of the 14th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '87, pages 163–169, New York, NY, USA, 1987.

[12] Jan Mattner, Sascha Lange, and Martin Riedmiller. Learn to swing up and balance a real pole based on raw visual input data. In *International Conference on Neural Information Processing*, pages 126–133. Springer, 2012.

[13] Eric Rohmer, Surya PN Singh, and Marc Freese. V-rep: A versatile and scalable robot simulation framework. In *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1321–1326. IEEE, 2013.

[14] Guido Schillaci, Verena V Hafner, and Bruno Lara. Exploration behaviors, body representations, and simulation processes for the development of cognition in artificial agents. *Frontiers in Robotics and AI*, 3:39, 2016.

[15] Evan Shelhamer, Parsa Mahmoudieh, Max Argus, and Trevor Darrell. Loss is its own reward: Self-supervision for reinforcement learning. *arXiv preprint arXiv:1612.07307*, 2016.

[16] Linda Smith and Michael Gasser. The development of embodied cognition: Six lessons from babies. *Artificial life*, 11(1-2):13–29, 2005.

[17] Herke Van Hoof, Nutan Chen, Maximilian Karl, Patrick van der Smagt, and Jan Peters. Stable reinforcement learning with autoencoders for tactile and visual data. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3928–3934. IEEE, 2016.

[18] Andy Zeng, Shuran Song, Stefan Welker, Johnny Lee, Alberto Rodriguez, and Thomas Funkhouser. Learning synergies between pushing and grasping with self-supervised deep reinforcement learning. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4238–4245. IEEE, 2018.