JAIST Repository
https://dspace.jaist.ac.jp/

Title	A 2-Stage Framework for Learning to Push Unknown Objects
Author(s)	Gao, Ziyan; Elibol, Armagan; Chong, Nak Young
Citation	2020 Joint IEEE 10th International Conference on Development and Learning and Epigenetic Robotics (ICDL-EpiRob)
Issue Date	2020-10
Type	Conference Paper
Text version	author
URL	http://hdl.handle.net/10119/17023
Rights	This is the author's version of the work. Copyright (C) 2020 IEEE. 2020 Joint IEEE 10th International Conference on Development and Learning and Epigenetic Robotics (ICDL-EpiRob), 2020, DOI:10.1109/ICDL-EpiRob48136.2020.9278075. Personal use of this material is permitted. Permission from IEEE must be obtained for all other uses, in any current or future media, including reprinting/republishing this material for advertising or promotional purposes, creating new collective works, for resale or redistribution to servers or lists, or reuse of any copyrighted component of this work in other works.
Description	

JAIST
JAPAN
ADVANCED INSTITUTE OF
SCIENCE AND TECHNOLOGY

Japan Advanced Institute of Science and Technology

A 2-Stage Framework for Learning to Push Unknown Objects

Ziyan Gao, Armagan Elibol, and Nak Young Chong

***Abstract*—Robotic manipulation has been generally applied to particular settings and a limited number of known objects. In order to manipulate novel objects, robots need to be capable of discovering the physical properties of objects, such as the center of mass, and reorienting objects to the desired pose required for subsequent actions. In this work, we proposed a computationally efficient 2-stage framework for planar pushing, allowing a robot to push novel objects to a specified pose with a small amount of pushing steps. We developed three modules: Coarse Action Predictor (CAP), Forward Dynamic Estimator (FDE), and Physical Property Estimator (PPE). The CAP module predicts a mixture of Gaussian distribution of actions. FDE learns the causality between action and successive object state. PPE based on Recurrent Neural Network predicts the physical center of mass (PCOM) from the robot-object interaction. Our preliminary experiments show promising results to meet the practical application requirements of manipulating novel objects.**

***Index Terms*—Robot Planar Pushing, Probabilistic Model, Data-Driven Approach**

I. Introduction

Robotic manipulation plays an important role in a wide range of industries and it is gradually advancing toward tackling skill-needed works, such as grasping or pushing novel objects, from predetermined and repetitive tasks. Pushing is a simple yet powerful action that can be used to rearrange objects to a working line or exploring the physical properties of objects such as the center of mass, inertia, and similar others. Besides, pushing action can be used to complement or replace grasping [1], allowing to achieve better grasping options through reorienting objects. For repositioning or re-orienting objects that are not easily grasped, such as the case that object size is larger than the maximum openness of the gripper, pushing gives an efficient way to change object state with smaller efforts than grasping.

In [2], it was 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. In the recent works [3] and [4], they proposed novel methods to predict intrinsic object properties such as the center of mass (COM) or inertia of the object. Inspired by [3], we proposed an efficient 2-stage robotic pushing framework with disentangled learning modules for enabling exploration of COM of the object and pushing it to a desired pose.

All authors are with the School of Information Science, Japan Advanced Institute of Science and Technology, Nomi, Ishikawa 923-1292, Japan {s1920013, aelibol, nakyoung}@jaist.ac.jp

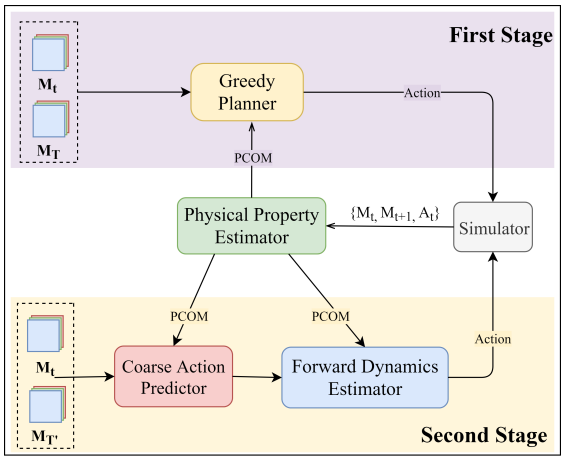

Fig. 1. The proposed 2-stage framework for robot planar pushing. In the first stage, a greedy planner utilizing PCOM of the current object and VCOM of the target object in the image plane minimizes relative position error. In the second stage, CAP and FDE find push actions for adjusting object pose to the sub-goal given the current object mask M_t and sub-goal object mask $M_{T'}$. PPE receives successive images and executed action recurrently to update its own internal cell state to predict PCOM, predicted PCOM to greedy planer in the first stage and to CAP and FDE in the second stage.

In the first stage, as shown in Fig. 1, a greedy planer is used to minimize the relative position error between the current position of the object and the target region. In the second stage, both object orientation and position are adjusted to the target pose. Physical Property Estimator (PPE) offers the physical center of mass (PCOM) of the current object state, which is an instinct property that cannot be measured directly from the input image of the object, both in the first and second stage. The greedy planner utilizes the predicted PCOM and the visual center of mass (VCOM) of the target to re-position the object. Please note that PCOM is the coordinates defined in the XY plane and transformed into the image plane. Z-direction is not considered. In this stage, the computational effort is small, since only PCOM is estimated, and there is no other computation involved. In the second stage, we used additional two modules: Coarse Action Predictor (CAP) and Forward Dynamics Estimator (FDE). CAP is a probabilistic model based on Mixture Density Network (MDN [5]), which predicts action distributions given by the current and target object masks, and PCOM of the current object state. Instead of random sampling in action space, we sample actions from a mixed Gaussian distribution given by CAP. By utilizing CAP, the number of sampling can be decreased dramatically. The

FDE module is a simple yet efficient plane neural network that utilizes PCOMs, VCOMs of the current object masks, and the executed action to predict the future state caused by these action candidates. During the second stage, we generate a sub-goal object mask $M_{T'}$ using interpolation between the current and target object masks similar to [3] to help the robot reorient or reposition the object.

We performed extensive simulations to evaluate our approach quantitatively. The experimental results show that the proposed framework samples efficiently for pushing tasks with the help of CAP. PPE helps other models to make a more accurate prediction to reorient objects, and FDE tremendously improves the performance of the proposed models.

II. Related Work

A data-driven approach for robot planar pushing was studied in [6], where a model was proposed to learn a function of object shape for pushing. However, the number of objects was limited and insufficient for performance evaluations. Recently, in [3], Long Short Term Memory [7] was utilized to learn future outcomes through historical experience of robot-object interaction, also based on Voting Theorem proposed by [8] which pointed out that with the help of PCOM, the rotation direction can be determined. They conducted auxiliary learning of PCOM. Even though this method releases many assumptions compared with [9], the proposed method still needs extensive action sampling (e.g., 1000 action samples are reported in the experiments). In this study, we developed a probabilistic model to learn the action distribution from collected data to reduce the number of sampling. In our experiments, sampling 50 action candidates are enough to perform on par with larger action sample sizes. Moreover, in the first stage of our framework, action sampling is not required. Thus, the computational cost can be further reduced.

Xu et al. [4] proposed a method that made use of several dynamic interactions (e.g., sliding or colliding) to predict dense representations that reflect the physical properties of the object. They developed a module called Multi-step aggregator to update the action-state representation in order to infer better dense representation. In this research, we use PPE, which consists of two LSTM layers and one output layer to predict PCOM. Since LSTM possesses input, output, forget gate mechanisms, it can efficiently propagate useful information through robot-object interaction.

Kloss et al. [10] investigated the advantages and limitations of neural network-based learning approaches for predicting the effects of actions based on sensory input. They showed how analytical and learned models could be combined to compensate each other. Baumeister et al. [11] combined analytical and learned models to a hybrid dynamic model for model predictive control. Gao et al. [12] proposed a combined prediction model (analytical and learned model) and an online learning framework for planar push prediction. However, these methods still hold several assumptions, such as known friction, which is challenging to obtain in a new case or need large computations. In this work, we try to develop an efficient model that should be good enough for selecting an action. Agboh et al. [13] used a deep learning model as a coarse model for Parareal to accelerate physics predictions and applied to Model Predictive Control. In this research, instead of the raw sensory input, we extract relevant features such as contact point, norm associated with a contact point, action velocity, predicted PCOM, and similar others. It should be noted that in order to reduce the gap between reality and simulation, we do not use the 'real' contact point and associated norm which can be obtained from the simulator. Instead, we extract the contact point and associated norm based on the geometric relationship by using Shapely [14]. We develop a fully connected neural network to predict the future outcomes of an action. It is computationally efficient and shows promising performance at action selection in our experiment.

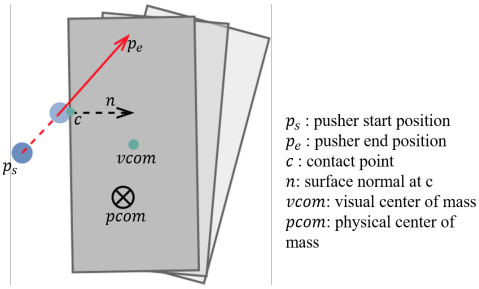

Fig. 2. Terminology for pushing.

III. 2-Stage Framework

A. Problem Statement

We consider the following problem: Given the initial and target object masks M_0 and M_T, the proposed models need to predict proper actions to re-arrange objects from the initial pose to the target pose. Before introducing the proposed models, firstly, we need to make a definition of action and terminology for pushing.

An action is defined by

$$a = [p_s, p_e], \tag{1}$$

where p_s consists of x_s and y_s for representing the initial horizontal and vertical coordinates of the pusher while p_e consists of x_e and y_e for representing the end of pushing coordinates, respectively.

The terminology for pushing is illustrated in Fig. 2. It should be noted that PCOM cannot be measured directly from an image, and VCOM is the visual center of mass of the object mask.

B. Coarse Action Predictor

The illustration of the proposed CAP is given in Fig. 3. The input to CAP is M_t and M_{t+1}, which are the current and sub-goal object masks (M_t,$M_{T'}$), and current PCOM predicted by PPE. $M_{T'}$ is generated by using interpolation between the current and target object mask. CAP predicts the distribution both for a start and terminating position of pushing

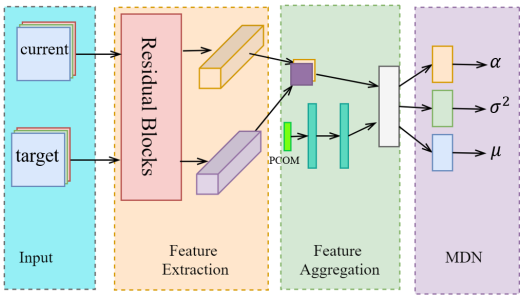

Fig. 3. Overview of Coarse Action Predictor (CAP) module

action. CAP consists of three parts: Feature Extraction, Feature Aggregation, and Mixture Density Network (MDN). We used residual blocks trained on ImageNet as Feature Extraction part. For Feature Aggregation part, it integrates feature maps of M_t and M_{t+1} obtained by Feature Extraction part, and a latent representation of PCOM obtained through two fully connected layers of size $32, 32$. MDN predicts action distribution parameterized by α, σ^2 and μ. The mixture distribution is computed by the following equation:

$$P(a_t|M_t, M_{t+1}, pcom_t) = \sum_{c=1}^{C} \alpha_c \mathcal{D}(\mu, \sigma^2), \qquad (2)$$

where c denotes the index of the corresponding mixture component. There are up to C mixture components, α_c is the coefficient of c component 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. After training CAP, we can sample action candidates from the predicted distribution, and this helps reduce the number of samples and computational costs.

C. Forward Dynamic Estimator

The FDE module is a plane neural network that has three layers of size 64, 64, and 3. ReLU activation function is added after every layer except for the output layer. The output is $[x_{t1}, y_{t1}, o_{t1}]$, where x_{t1}, y_{t1} are the future object position and o_{t1} is the future object orientation. FDE is a function of a low-dimensional feature vector shown in Eq. 3, where c, n represent the contact position in the global frame and associated surface norm, while A represents the shape area. The output is the future movement of the object represented by O_{t+1} in Eq. 3. O_{t+1} composed of the future position and orientation of the object in the image frame. These input features have a direct effect on object movement. $vcom$ infers the object position and $pcom$ can help FDE predict the rotation direction. c, n can help determine the type of contact: "sticking contact" or "sliding contact". In the first case, the object movement will be the same as the velocity of the pusher, while in the second case, the object movement will be almost orthogonal to the resulting motion at the contact point. A can be seen as a factor implicitly encoding the contact friction between the object and the plane.

$$O_{t+1} = f_{FDE}(a_t, c_t, n_t, vcom_t, pcom_t, A) \qquad (3)$$

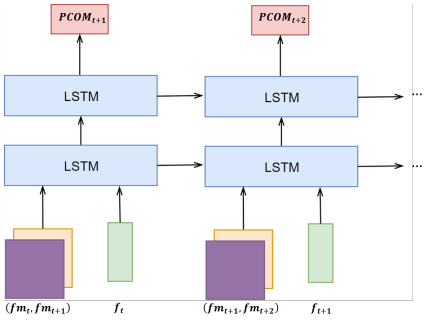

Fig. 4. Illustration of Physical Property Estimator, fm_t, fm_{t+1} are the feature maps extracted from successive object mask, f_t is the low-dimensional vector.

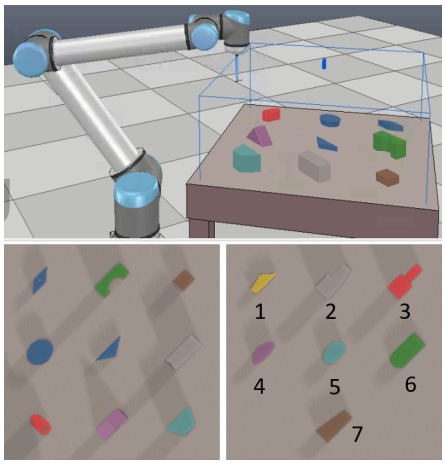

Fig. 5. Simulation Environment: nine objects in data collection phase (left lower) and seven novel objects (right lower) in evaluation phase. The size of novel objects is ranging from $14cm^2$ to $60cm^2$.

D. Physical Property Estimator

PPE is modeled as a recurrent neural network, and it is composed of two LSTM layers of size 64 each and 2 plane layers of size 32 and 2. ReLU activation function is added after each layer except the output layer. The input to PPE has two parts; one is two feature maps of two successive object masks obtained by convolution neural network that is the same as Feature Extraction part of the CAP module and the other one comes from a low-dimensional feature vector shown in Eq. 4, where $pcom_{t+1}$ is the predicted PCOM of the object shown in the second object mask.

$$pcom_{t+1} = f_{PPE}(fm_{t:t+1}, vcom_{t:t+1}, c_t, n_t, a_t) \qquad (4)$$

IV. Experimental Settings

All models are implemented in Pytorch [15]. The dataset is collected through extensive simulations done using the CoppeliaSim robot simulator. A cylinder with a radius of $0.5cm$ and length $20cm$ is attached to the end link of the UR10 robot. A square table of $0.6m$ is used as the working space for pushing. A camera is mounted on the ceiling above the table. Nine objects (depicted in Fig. 5) of different shapes and sizes were considered in the experiments. For the sake of stability,

PCOM is aligned to the object randomly within the half size of its bounding box. In order to improve the capabilities of the model, we randomly change the object size and the ratio of its dimensions. Firstly, an object is loaded into the scene with a random pose, then an action, which is performed as a straight line of length $2.5cm$, is sampled to interact with the object. Actions are selected randomly with a guarantee that each action changes the object pose. The procedure was repeated multiple time steps. Finally, we record the object mask and the executed action in each time step. The dataset contains more than 57,000 sequences of interactions between the robot and different objects. The lengths of these sequences are ranging from 3 to 8. In order to re-use residual network layers, we tiled mask images to three channels.

In the training phase of CAP, we used the negative log-likelihood function to minimize the training error.

$$L_{CAP} = -log(P(a_t|M_t, M_{t+1}, pcom_t)) \tag{5}$$

We used the mean square error to minimize the training error of FDE. For PPE, we used the weighted mean square error defined in Eq. 6. The intuition behind this loss function is that, with the increase of the number of interaction, the accuracy of PCOM estimation in the current step should be higher than the previous step.

$$L_{PPE} = \frac{1}{T}\sum_{t=1}^{T} \alpha_t (\hat{pcom}_t - pcom_t)^2, \alpha_t = \frac{t}{\sum_{t=1}^{T} t} \tag{6}$$

For the training of all modules, we used Adam optimizer. We set the size of mini-batch to 64 for training CAP and PPE, and 200 for training FDE. The learning rate was set to 0.001 for CAP and PPE, and 0.0001 for FDE. We added the Dropout layer before each output layer for all models and set the deactivate ratio to 0.1. In the whole training phase, the parameters of residual blocks were fixed.

In the evaluation phase, we aim to analyze the proposed 2-stage method for robot planar pushing both in terms of efficiency and accuracy. We conducted two main experiments. For the first experiment, we try to answer the question: How many action candidates are needed to complete the pushing task? For the second experiment, we try to answer the question: how do PPE and FDE modules contribute to the pushing task?

In the first experiment, We expect that with the number of sampling increases, the accuracy will improve, and the average number of pushing steps will decrease. Firstly, an object is loaded into the workspace, and then we use a greedy planner to minimize relative position error. After that, we use the CAP and FDE modules to select an action to push the object to the target pose. Action candidates are sampled from the predicted action distribution, and action is selected based on the mean squared error calculated between the required change $O_{T'}$ and $\hat{O}_{t+1}$ predicted by FDE. In this experiment, both initial and target object positions are sampled randomly in a range of [-0.28m, 0.28m], and the relative orientation error between the initial and target is set to 180 degrees. The range of [-0.28m, 0.28m] is selected because we want to keep the objects on the table. Compared with the evaluation settings in [3], where the relative orientation between initial and target pose is sampled between [-90,90] degrees, our case is more challenging. Specifically, the number of steps executed in the first stage is not taken into account, and only the pushing steps in the second stage are counted, since the greedy planner in the first stage can re-position the object successfully in all new cases. In the second stage, we generate a sub-goal that has $2.5cm$ and 17 degrees closer to the target pose than the current pose, since the pushing length of $2.5cm$ is adopted during data collection. Both $2.5cm$ and 17 degrees can be changed. However, a small orientation change will cause more steps to reach the target pose, and a massive orientation change will cause oscillations in orientation. We tried different orientations of required change several times and found that the 17 degree case provides a good compromise between efficiency and stability. The robot needs to push the object from the initial pose to the target pose within 20 steps. In the second stage, if the final pose of the object is close to the target pose (for the relative position error, it should be within $\pm 0.05m$, for the relative orientation error, it should be within $\pm$10degree, the same criteria used in [3]), we regard this trial as success. We used action candidates of 3, 10, 50, 100, 200 to 500 sampled from the action distribution predicted by CAP and keep other settings the same. For each setting, we repeat the same experimental trial for 100 times.

For the second experiment, we investigate the pure-rotation and pure-translation actions. In this experiment, we only use the modules in the second stage. PCOM is randomly aligned to the object similarly in the first experiment. For the pure-rotating object experiment, the object position is not taken into account, and it was set to the center of the table with a random orientation. The robot is expected to rotate objects 17 degrees at each time step. The robot needs to rotate objects 180 degrees within the relative orientation error ± 10 degrees within 20 steps. For a pure-translating object, the object position is set to the center of the table as the target position. The initial position is set randomly in a range of [-0.28m, 0.28m] and the initial and target orientation were set to remain the same. The robot is expected to re-position the object $2.5cm$ at each time step and keep its orientation unchanged. The robot needs to move an object to the target region within the relative orientation error ± 10 degrees. For each experiment, we used three alternative configurations based on proposed models:

- **Test Model 1** Model with a known PCOM; Instead of relying on PCOM predicted by PPE, we directly provide PCOM to CAP and FDE. Because of the precise PCOM information, this setting should have higher performance compared with other settings.
- **Test Model 2** Model without the PPE module; Instead of predicting PCOM, the setting assumes that PCOM overlaps VCOM
- **Test Model 3** Model without the FDE module; Instead of selecting the best action from sampled action candidates

TABLE I
ACCURACY FOR THE FIRST EXPERIMENT

Number of Samples	3	10	50	100	200	300	400	500
object1	0.58	0.79	**0.97**	0.96	0.96	0.91	0.96	**0.97**
object2	0.52	0.86	0.98	0.98	0.97	**0.99**	**0.99**	0.97
object3	0.49	0.81	0.98	0.98	**1.0**	0.99	0.99	0.99
object4	0.55	0.86	0.98	1.0	0.95	**0.99**	0.98	0.97
object5	0.44	0.92	0.98	0.98	0.97	0.99	**0.99**	0.98
object6	0.55	0.92	0.98	0.99	0.99	**1.0**	0.99	**1.0**
object7	0.56	0.86	0.94	0.96	0.98	**0.98**	0.97	0.97

TABLE II
ACCURACY FOR PURE ROTATING OBJECTS

Models	object1	object2	object3	object4	object5	object6	object7
Test Model1	0.99	**1.0**	0.98	**0.98**	**1.0**	0.99	**0.98**
Proposed Model	0.98	0.99	**1.0**	0.97	0.96	**1.0**	**0.98**
Test Model2	**1.0**	0.97	**1.0**	0.94	0.99	0.98	0.95
Test Model3	0.23	0.12	0.15	0.24	0.16	0.21	0.22

with the help of FDE, the setting select a best action randomly from the predicted action candidates.

We conducted 100 trials for each novel object and each testing configuration. For a fair comparison, we use the same initial and target object pose and the same PCOM position relative to the object frame in the same trial for all settings.

V. EXPERIMENTAL RESULTS AND DISCUSSION

The result of our experiments are provided in Figures 7, 8, 9, and Tables I, II, and III. In these figures, we show the box plot of pushing steps for successful pushing, where we use the 10th and 90th quantile to represent the min and max pushing step. In these tables, we report the accuracy for pushing objects from the initial to desired pose calculated by Eq. 7, where $N_{success}$ is the number of successful trials and N_{total} is the number of trials conducted. For all experiments, N_{total} is 100.

$$acc = \frac{N_{success}}{N_{total}} \tag{7}$$

The result of the first experiment is presented in Fig. 7 and Tab. I. We observe from the results that the average number of steps has decreased gradually, especially from the settings of 3 action samples to 50 action samples. From 50 action samples to 500 action samples, no such improvement was found as in the case of smaller numbers of samples. For the accuracy, we found that there is a significant improvement from the setting of sampling 3 actions to sampling 50 actions, and there was no considerable improvement in accuracy for other settings.

The results of the second experiment are summarized in Fig. 8, Fig. 9 and Tab. II, Tab. III. From Fig. 8 and Tab. II, we observed that Test Model 3 performed worse than others, and Test Model 1 performed best for almost every novel object. In the experiment of pure rotation, Test Model 2 and the Proposed Model performed similarly in terms of the mean pushing steps. However, we observed that the pushing steps of Test Model 2 have varied greatly compared with the Proposed Model. In the experiment of pure translation, we observed that the pushing steps of Test Model 2 have varied less compared with the Proposed Model.

Based on the experimental results and observations, we found that our proposed framework is efficient in action sampling, and furthermore it uses much fewer action samples to yield promising performance. In the second experiment, firstly, we found that FDE has a significantly important role in action selection, as it can be seen in the case of Test Model 3 which does not utilize FDE, showing much worse performance compared with other settings. Besides, we observed that Test Model 1, in which we directly provide the ground truth PCOM, performed almost best in terms of pushing steps. Therefore, with the advantage of known PCOMs, the other module can complete the pushing task with a smaller number of pushing steps, especially in the case of a pure rotation experiment.

In the pure translation experiment, we still can find that Test Model 1 and the Proposed Model have slightly better performance compared with Test Model 2, even though the differences are not as much as the ones in the pure rotation experiment. This might be due to the fact that not only PCOM but also object shape and initial position have a large impact on the performance of our models. Fig. 6 shows several examples of our pure-translation experiment.

We carried out an additional experiment for evaluating how PCOM affects the performance of FDE. We used the test data of the data set collected in IV. We used two input features for comparison: features containing PCOM and VCOM, respectively. For the second input feature, we simply replace PCOM by VCOM. We use the mean square error to compute the predicted position error and mean absolute error to compute the predicted orientation error. The result is shown in Fig.10. We can see that there is no obvious difference between the two input features in terms of position prediction error. However, PCOM does help FDE predict the orientation of the object when calculating the orientation prediction error. We can see that FDE with PCOM has a smaller prediction error and lower variance than FDE with VCOM.

TABLE III
ACCURACY FOR PURE TRANSLATING OBJECTS

Models	object1	object2	object3	object4	object5	object6	object7
Test Model1	0.92	0.97	**1.0**	**0.97**	0.97	0.99	0.98
Proposed Model	0.98	0.98	0.99	**0.97**	**0.98**	**1.0**	0.99
Test Model2	0.98	**0.99**	**1.0**	0.95	0.97	**1.0**	**1.0**
Test Model3	0.49	0.70	0.66	0.54	0.56	0.69	0.48

VI. CONCLUSIONS AND FUTURE WORK

In this study, we proposed a 2-stage framework with disentangled learning modules for robot planar pushing. Our proposed framework allows exploring PCOM and pushing a wide variety of novel objects given a desired pose. The validity of the framework was evaluated quantitatively through extensive simulations. The obtained results show that our proposed method can manipulate novel objects with unknown physical properties. Compared with previous work, our method is efficient in sampling and can handle more challenging tasks. Through the experiments, our method achieved promising accuracy by only utilizing 50 action samples instead of 1000, as reported in [3]. Furthermore, we found that the proposed

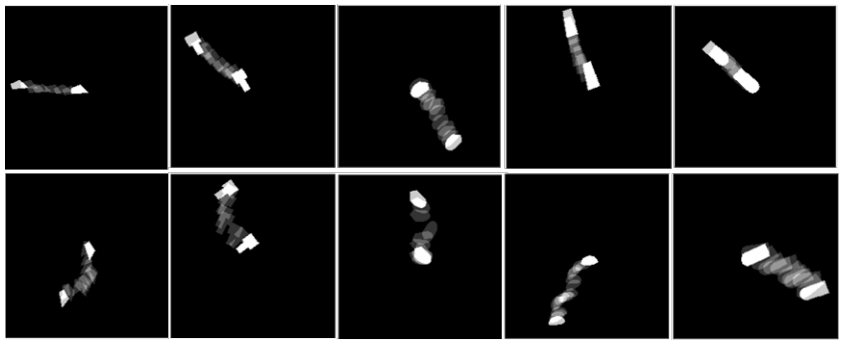

Fig. 6. Examples of a pure-translating object, in the first row, objects are pushed without a significant change in orientation. In contrast, in the second row, the orientation of the object varies in a large interval.

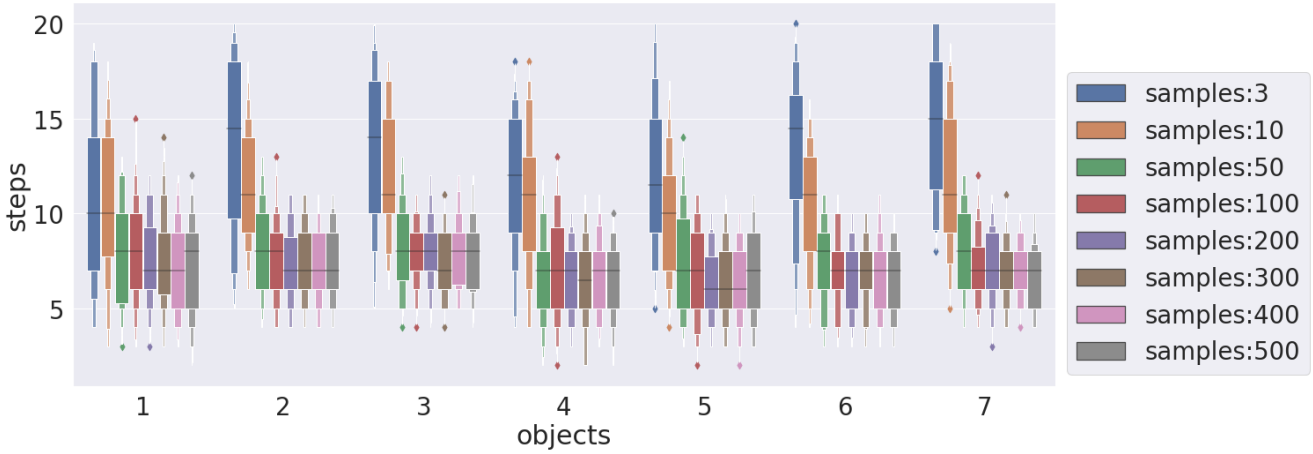

Fig. 7. Box plots of the pushing steps for re-orienting objects to the desired pose. The vertical axes represent the steps executed to successfully re-orient the objects. The horizontal labels are the object categories. For each object, there are multiple corresponded box plots, which represent a different number of action candidates sampled from the output of CAP. We used 10th and 90th quantile to represent the min and max step, respectively.

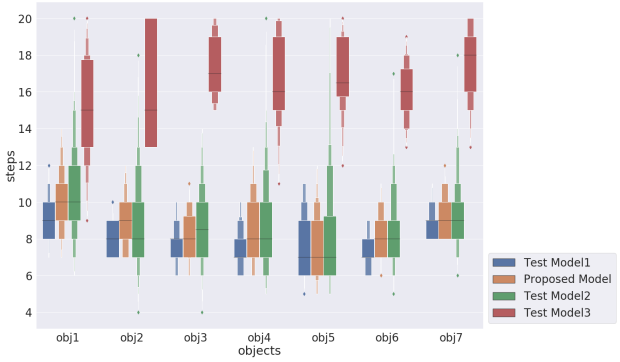

Fig. 8. Box plots of the pushing steps for rotating object 180 degrees without considering the object position. The vertical axes represent the steps to successfully rotate the objects. The horizontal labels are the object categories. For each object, there are multiple corresponded box plots, which represent different models used.

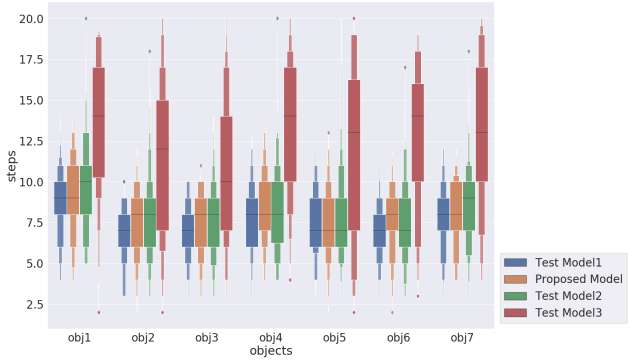

Fig. 9. Box plots of the pushing steps for translating object without considering the object orientation. The vertical axes represent the steps to successfully translate the objects. The horizontal labels are the object categories. For each object, there are multiple corresponded box plots, which represent different models used.

PPE does help the framework yield improved predictions for pushing novel objects, and the proposed FDE plays a pivotal role in action selection.

Despite the competitive advantages of the proposed framework, it has not been yet evaluated on the real platform, which will be our future work. In addition, in our settings, the shapes of objects used were relatively simple and similar to each other. We will test more complex shaped objects to further evaluate our model. We also used a cylinder as the pushing

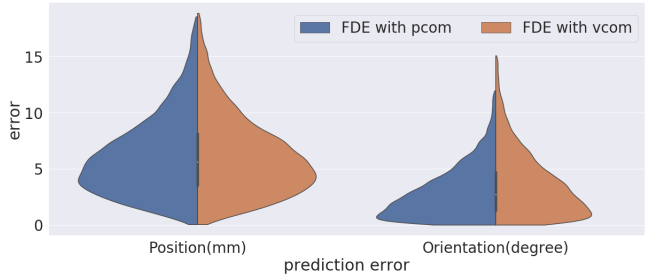

Fig. 10. Evaluation for Forward Dynamic Estimator; The mean of position prediction error for FDE with PCOM and FDE with VCOM is 6.29 and 6.30 millimeters. The orientation prediction error for FDE with PCOM and FDE with VCOM is 3.03 and 3.77 degrees.

tool to obtain precise contact points. Considering the different types of widely used tools such as parallel-jaw gripper will also be a future research direction.

REFERENCES

[1] 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.

[2] 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.

[3] Juekun Li, Wee Sun Lee, and David Hsu. Push-Net: Deep Planar Pushing for Objects with Unknown Physical Properties. 2018.

[4] Zhenjia Xu, Jiajun Wu, Andy Zeng, Joshua Tenenbaum, and Shuran Song. DensePhysNet: Learning Dense Physical Object Representations Via Multi-Step Dynamic Interactions. 2019.

[5] Christopher M Bishop. Mixture density networks. 1994.

[6] Tucker Hermans, Fuxin Li, James M Rehg, and Aaron F Bobick. Learning contact locations for pushing and orienting unknown objects. In *2013 13th IEEE-RAS international conference on humanoid robots (humanoids)*, pages 435–442. IEEE, 2013.

[7] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Comput.*, 9(8):1735–1780, November 1997.

[8] Matthew T. Mason. Mechanics and Planning of Manipulator Pushing Operations. *The International Journal of Robotics Research*, 5(3):53–71, 1986.

[9] Jiaji Zhou, Robert Paolini, J Andrew Bagnell, and Matthew T Mason. A convex polynomial force-motion model for planar sliding: Identification and application. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 372–377. IEEE, 2016.

[10] Alina Kloss, Stefan Schaal, and Jeannette Bohg. Combining learned and analytical models for predicting action effects. 2017.

[11] Thomas Baumeister, Alina Kloss, and Jeannette Bohg. Combining Analytical and Learned Models for Model Predictive Control. (Nips), 2018.

[12] Huidong Gao, Yi Ouyang, and Masayoshi Tomizuka. Online learning in planar pushing with combined prediction model. *arXiv preprint arXiv:1910.08181*, 2019.

[13] Wisdom C Agboh, Daniel Ruprecht, and Mehmet R Dogar. Combining coarse and fine physics for manipulation using parallel-in-time integration. *arXiv preprint arXiv:1903.08470*, 2019.

[14] Sean Gillies et al. Shapely: manipulation and analysis of geometric objects, 2007–.

[15] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. 2017.